Optimasi Kasiski Examination pada Studi Kasus SPOJ The Bytelandian Cryptographer (Act IV)

Freddy Hermawan Yuwono, Rully Soelaiman dan Wijayanti Nurul Khotimah
Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember
Jl.Teknik Kimia, Kampus ITS Sukolilo, Surabaya 60111 Indonesia
e-mail: freddy.yuwono@gmail.com, rully@is.its.ac.id, wijayanti@if.its.ac.id

***Abstrak*—Pada era digital ini, tingkat kebutuhan masyarakat akan informasi semakin meningkat. Hal ini menyebabkan pertukaran informasi menjadi sangat mudah. Hal ini membuat informasi yang bersifat sensitif dapat terjadi kebocoran informasi kepada pihak-pihak yang tidak berkepentingan. Kebocoran informasi terbagi menjadi dua apabila dilihat dari keutuhan informasi yang didapat, yaitu sebagian dan seutuhnya. Kebocoran informasi yang bersifat sebagian, membuat pihak-pihak yang tidak berkepentingan tetapi yang menginginkan informasi tersebut, berusaha untuk mendapatkan informasi yang utuh dari potongan-potongan informasi yang telah didapatkan.**

Permasalahan dalam artikel ini adalah permasalahan untuk mendapatkan *plaintext* sebanyaknya dari *ciphertext* dan batas atas panjang kunci yang telah diketahui. Metode enkripsi yang digunakan merupakan teknik *Vigenere Cipher*. Dalam permasalahan ini diberikan *plaintext* dan *ciphertext*, akan tetapi terdapat informasi yang hilang pada keduanya. Batas atas panjang kunci yang diberikan belum tentu panjang kunci yang sesungguhnya. *Plaintext* yang dapat direkonstruksi dari kepingan informasi yang telah didapatkan. Perlu untuk mencari panjang kunci dengan cara memodifikasi *Kasiski Examination* dan *Intersection*. Beberapa hal yang perlu diperhatikan seperti mempercepat *Kasiski Examination* dan juga *Intersection*.

Hasil dari metode ini telah berhasil untuk menyelesaikan permasalahan yang telah diangkat dengan benar. Waktu yang diperlukan untuk dapat menyelesaikan masukan sebesar 2MB rata-rata dalam 4,42 detik dengan alokasi memori sebesar 26,5MB.

***Kata Kunci*—Ciphertext, Kasiski Examination, Optimisasi, Plaintext.**

I. PENDAHULUAN

Informasi yang ada tidak terlepas dari kebutuhan manusia mengenai keadaan di sekitarnya. Informasi yang diterima seseorang pada masa sekarang dapat melalui media fisik dan media digital. Media fisik seperti koran dan majalah, sedangkan media digital seperti Facebook dan Twitter. Media-media tersebut sanggup untuk menyebarkan informasi dengan sangat cepat, sehingga orang-orang dengan cepat mengetahui informasi yang berada disekitarnya. Informasi digital yang beredar di dunia maya pun tidak lepas dari penyalahgunaan informasi. Dibutuhkan suatu teknik penyandian terhadap informasi yang dimiliki agar informasi yang bersifat rahasia itu tidak diketahui dengan orang–orang yang tidak berkepentingan. Akan tetapi hal ini menarik perhatian dari pihak-pihak yang menginginkan informasi tersebut. Informasi yang diperoleh hanya terbatas dengan kepingan-kepingan informasi saja. Seperti contohnya studi kasus SPOJ *The Bytelandian Cryptographer (Act IV)*. Pada studi kasus ini diketahui bahwa metode enkripsi yang digunakan adalah *Vigenere Cipher*. Pada studi kasus diberikan sejumlah kasus uji coba, untuk setiap kasus uji coba diberikan sejumlah potongan-potongan informasi dari *plaintext* dan *ciphertext* dengan batas atas dari panjang kunci yang digunakan. Panjang Kunci yang diberikan bukan panjang kunci yang sesungguhnya. Diharapkan dari studi kasus tersebut adalah merekonstruksi ulang *plaintext* dari data yang telah tersedia. Batasan masalah jumlah input tidak akan melebihi dari 2 MB. Batas atas panjang kuncinya bernilai $1 \leq M \leq 100.000$, dan jumlah kasus ujicoba $1 \leq T \leq 200$ [1]. Solusi dari studi kasus yang akan dibahas dan diimplementasikan dalam bentuk kode.

II. TINJAUAN PUSTAKA

A. Kasiski Examination

Kasiski Examination adalah suatu teknik yang digunakan untuk mencari panjang kunci dari *ciphertext* yang berasal dari teknik enkripsi *Polyalphabetic Cipher* dan turunannya. *Kasiski Examination* memanfaatkan kelemahan dari *Polyalphebetical Cipher* tanpa harus mengetahui *plaintext* dan himpunan kunci yang digunakan. Kelemahannya yaitu apabila suatu *substring plaintext* yang sama dienkripsi dengan *substring* dari kunci yang digunakan akan menghasilkan pola yang sama [2]. Metode pencarian panjang kunci yang dilakukan adalah sebagai berikut:

1) Mencari semua substring yang berulang pada suatu kalimat.
2) Mencari panjang di mana subtring tersebut berulang kembali.
3) Mencari semua faktor dari nilai yang diperoleh dari tahap dua.
4) Mencari faktor persekutuan terbesar dari hasil yang diperoleh pada tahap tiga.

B. Intersection

Intersection adalah himpunan *A* dan himpunan *B*, di mana ada bagian dari *A* juga merupakan bagian dari *B* [3]. Sebagai contoh *intersection* antara {1,2,3} dan {1,4,5} adalah {1}. *Intersection* pada artikel ini digunakan untuk mengeliminasi elemen yang sama yang telah dihasilkan oleh *Kasiski Examination*.

III. METODE PENYELESAIAN

Dalam bagian pendahuluan telah dijelaskan bahwa enkripsi yang digunakan adalah *Vigenere Cipher*. Enkripsi yang dilakukan mengikuti aturan Persamaan 1.

$$y_i = x_i + k_{1+(i-1) mod\ M}\ Mod\ 26 \quad (1)$$

Dengan y_i adalah fungsi enkripsi pada indeks ke-i, x_i adalah *plaintext* pada indeks ke-i, k_1 adalah himpunan kunci pada indeks ke-1, M adalah panjang himpunan kunci. Pada studi kasus input yang bernilai A sampai dengan Z, dapat direpresentasikan menjadi 0 sampai dengan 25. Tahapan-tahapan untuk menyelesaikan studi kasus ini sebagai berikut:

a) Menyimpan posisi indeks karakter, di mana pada indeks tersebut baik *ciphertext* maupun *plaintext* tidak bernilai "*", beserta menyimpan hasil perhitungan selisih antara *ciphertext* dan *plaintext* [4].
b) Menyimpan posisi indeks, apabila *ciphertext* diketahui dan *plaintext* tidak diketahui [4].
c) Pada tahapan ini adalah modifikasi dari *Kasiski Examination*, apabila menggunakan *Kasiski Examination* pada umumnya yang hanya mencari posisi berulang dari suatu *substring* dalam suatu kalimat tidak bisa digunakan untuk menyelesaikan permasalahan ini. Oleh karena itu diubah menjadi mengiterasi panjang kunci dari 1 sampai dengan *M*, yang akan digunakan untuk membagi selisih yang diketahui antara *plaintext* dan *ciphertext* sebesar posisi iterasi yang telah berjalan. Tahapan selanjutnya melihat dalam blok-blok yang telah terbentuk ini terdapat *collision* dan indeks yang saling bertabrakan memiliki nilai yang sama atau tidak. Apabila sama maka tidak terjadi tabrakan. Sebaliknya jika terjadi tabrakan maka harus mencari panjang kunci yang baru. Panjang kunci dapat dicari dari $\frac{M}{2} + 1 \leq N \leq M$. Panjang kunci dapat dicari dari $\frac{M}{2} + 1$ tidak dari 1 karena apabila suatu panjang kunci bernilai benar maka kelipatan dari panjang kunci itu pun juga pasti benar dan untuk mempersingkat waktu eksekusi yang seharusnya terjadi. Pada bagian ini dilakukan untuk mencari panjang kunci yang benar dengan cara mengiterasi hasil yang diperoleh pada tahap satu *modulo* dengan posisi iterasi yang dilakukan. Apabila tidak terjadi konflik maka panjang kunci tersebut benar. Jika sebaliknya yang terjadi maka panjang kunci tersebut salah. Pada bagian ini memungkinkan bahwa bisa jadi lebih dari satu panjang kunci yang bernilai benar [4].
d) *Intersection* yang dilakukan berada didalam perulangan panjang kunci pada waktu *generate* setiap karakter yang terdapat dalam penyimpanan tahap dua pada panjang kunci tersebut dengan ketentuan sebagai berikut:
    a) Panjang kunci harus benar.
    b) Apabila terdapat indeks yang tidak dapat dipastikan isinya maka posisinya harus dibuang dari penyimpanan dan hasilnya pasti "*".
    c) Apabila *plaintext* bernilai "*" dan himpunan kunci yang telah terbentuk tidak kosong pada indeks tersebut, maka *plaintext* akan bernilai sesuai dengan kunci yang terbentuk pada indeks tersebut.\

Pseudocode yang digunakan dalam menyelesaikan studi kasus ini adalah sebagai berikut:

a) Fungsi SOLVE
   Pada fungsi ini dilakukan *intersection* dari hasil yang didapatkan dari *Kasiski Examination*. *Pseudocode* fungsi dapat dilihat pada Gambar 1.
b) Fungsi VALIDITY
   Pada fungsi ini dilakukan *Kasiski Examination*. *Pseudocode* fungsi dapat dilihat pada Gambar 2.

```
1: function SOLVE(message,chiper,m)
2:     counter_diketahui ← 0
3:     counter_yang_ingin_diketahui ← 0
4:     diketahui[ ]
5:     Selisih_diketahui[ ]
6:     ingin_diketahui[ ]
7:     Key[ ]
8:     for i = 0 to message[i] ≠ 0 ; i+ = 1 do
9:         if message[i] ≠' *' dan cipher[i] ≠' *' then
10:            diketahui[counter_diketahui] = i
11:            Selisih_diketahui[i] = (message[i] − cipher[i] + 26)%26
12:            counter_diketahui = counter_diketahui + 1
13:        else if message[i] =' *' dan cipher[i] ≠' *' then
14:            ingin_diketahui[counter_yang_ingin_diketahui] = i
15:            counter_yang_ingin_diketahui + 1
16:        end if
17:    end for
18:    m = min(m, panjang message)
19:    for n = m/2 + 1 to n ≤ m;n+ = 1 do
20:        if VALIDITY(Key, counter_diketahui, diketahui, Selisih_diketahui, n) = True
   then
21:            counter ← 0
22:            while counter ≠ sizeof(ingin_diketahui) do
23:                if Key[ingin_diketahui[counter]%n] = null then
24:                    message[ingin_diketahui[counter]] =' *'
25:                    remove element index i in ingin_diketahui
26:                else if message[ingin_diketahui[counter]] =' *' then
27:                    message[ingin_diketahui[counter]] = (ciphertext[ingin_diketahui[counter]]−
   Key[ingin_diketahui[counter] + 26)%26
28:                    counter = counter + 1
29:                else if message[ingin_diketahui[counter]] ≠
   (ciphertext[ingin_diketahui[counter]] − Key[ingin_diketahui[counter] + 26) %26) then
30:                    message[ingin_diketahui[counter]] =' *'
31:                    remove element index i in ingin_diketahui
32:                else
33:                    counter = counter + 1
34:                end if
35:            end while
36:        end if
37:    end for
38:    puts(message)
39: end function
```

Gambar 1. *Pseudocode* fungsi SOLVE

```
1: function VALIDITY(Key,counter_diketahui,diketahui,Selisih_diketahui,n)
2:     Initialize(Key, −1)
3:     for i = 0 to i < counter_diketahui; i+=1 do
4:         temp = diketahui[i]
5:         if Key[temp%n] = −1 then
6:             Key[temp%n] = Selisih_diketahui[temp]
7:         else if Key[temp%n] ≠ Selisih_diketahui[temp] then return False
8:         end if
9:     end for
10:    return True
11: end function
```

Gambar 2. *Pseudocode* fungsi VALIDITY

Tabel 1. Kecepatan Maksimal, Minimal, dan Rata-Rata dari Hasil Uji Coba Pengumpulan 30 Kali pada Daring Pengujian SPOJ

Waktu Maksimal	4,49 detik
Waktu Minimal	4,38 detik
Waktu Rata-Rata	4,418 detik
Memori Maksimal	27 MB
Memori Minimal	26 MB
Memori Rata-Rata	26,5 MB

Kompleksitas dari fungsi SOLVE adalah $O((N+S)*\frac{M}{2}*(N+S))$ dan kompleksitas fungsi VALIDITY adalah $O(S)$. Sehingga kompleksitas yang terbentuk $O((N+S)+\frac{M}{2}*(N+2S))$. Kompleksitas akhir dari kedua fungsi tersebut adalah $O(\frac{M}{2}*(N+S))$ untuk setiap kasus uji yang diberikan. Dengan $\frac{M}{2}$ adalah jumlah batas atas kunci dibagi dengan 2, *N* adalah jumlah posisi karakter yang terdapat pada tahap 2, dan S adalah jumlah posisi karakter pada tahap 1.

IV. UJI COBA DAN ANALISIS

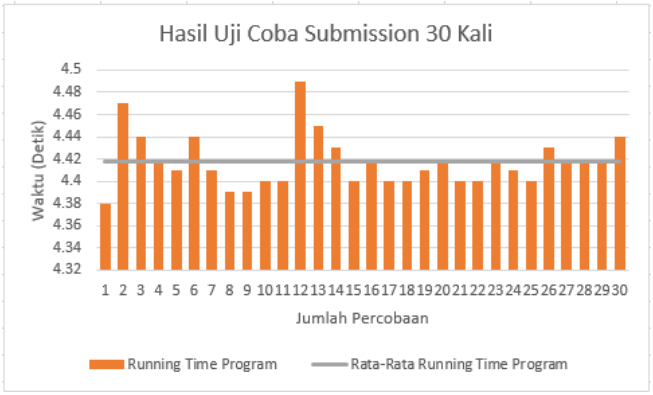
A. *Uji Coba Kebenaran*

Uji coba kebenaran dilakukan dengan cara mengumpulkan berkas kode implementasi ke dalam daring penilaian online SPOJ. Studi kasus yang diselesaikan adalah *The Bytelandian Cryptographer (Act IV)* dengan *code* soal CRYPTO4[1]. Hasil uji kebenaran dan waktu eksekusi program pada situs SPOJ ditunjukkan pada Gambar 3.

Uji coba kinerja dari implementasi program yang dihasilkan dengan cara mengumpulkan berkas kode implementasi ke dalam daring penilaian online SPOJ sebanyak 30 kali dengan mencatat waktu dan memori yang dibutuhkan. Dapat dilihat pada Gambar 4 dan Tabel 1.

20997300	2018-01-17 00:32:49	freddy	accepted	4.42	26M

Gambar 3. Hasil uji kebenaran dengan melakukan *submission* ke situs penilaian daring SPOJ



Gambar 4. Hasil Uji Coba *Submission* ke situs penilaian daring SPOJ sebanyak 30 kali

[1] http://www.spoj.com/problems/CRYPTO4/

Berdasarkan Tabel 1, dari percobaan yang telah dilakukan, didapatkan waktu eksekusi rata-rata 4,418 detik dan waktu maksimal sebesar 4,49 detik pada daring SPOJ.

B. *Analisis Kompleksitas*

Kompleksitas yang diperlukan untuk melakukan *Kasiski Examination* adalah $O(S)$. *Intersection* yang dibutuhkan untuk mengeliminasi hasil yang telah diperoleh dari *Kasiski Examination* adalah $O((N+S)*\frac{M}{2}*(N+S))$. Kompleksitas akhir adalah $O(\frac{M}{2}*(N+S))$ untuk setiap kasus uji. Algoritma *naïve* yang digunakan hampir sama dengan algoritma *Kasiski Examination* dan *intersection* yang dilakukan. Bedanya terlatak pada tahap *intersection* yang terdapat pada algoritma *naïve* dilakukan setelah semua perulangan pada algoritma *Kasiski Examination*. Algoritma *naïve* memiliki kompleksitas akhir $O(\frac{M}{2}*(N+S)^2)$ untuk setiap kasus uji. Sehingga dapat disimpulkan bahwa algoritma *Kasiski Examination* dengan *intersection* jauh lebih cepat daripada algoritma *naïve.*

V. KESIMPULAN DAN SARAN

Dari hasil uji coba yang telah dilakukan terhadap perancangan dan implementasi algoritma untuk menyelesaikan studi kasus SPOJ *The Bytelandian Cryptographer (Act IV)* dapat diambil kesimpulan sebagai berikut:

1) Implementasi algoritma dengan menggunakan teknik *Kasiski Examination* dengan adanya optimasi tidak dapat menyelesaikan permasalahan SPOJ *The Bytelandian Cryptographer (Act IV)* dengan benar. Dengan adanya metode *Intersection* yang dilakukan setelah teknik *Kasiski Examination* dengan optimasi dapat menyelesaikan studi kasus tersebut dengan benar.
2) Kompleksitas waktu $O(\frac{M}{2}*(N+S))$ untuk setiap kasus uji masih dapat menyelesaikan permasalahan SPOJ *The Bytelandian Cryptographer (Act IV)*.
3) Waktu yang dibutuhkan oleh program untuk menyelesaikan SPOJ *The Bytelandian Cryptographer (Act IV)* minimum 4.38 detik, maksimum 4.49 detik dan rata-rata 4.418 detik. Memori yang dibutuhkan berkisar antara 26-27 MB.

Saran yang dapat diambil dari metode yang telah di bahas sebagai berikut:

1) Teknik *Kasiski Examination* masih cenderung lambat. Hal tersebut terjadi karena masih menggunakan teknik *brute force* sehingga hasil yang diperoleh kurang optimal. Perlu adanya optimisasi lanjutan yang dapat mencari suatu panjang kunci.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan penelitian ini. Penulis juga mengucapkan

terima kasih kepada orang tua dan keluarga penulis, Bapak Rully Soelaiman dan Ibu Wijayanti Nurul K, selaku dosen pembimbing penulis dan kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama pengerjaan penelitian ini.

DAFTAR PUSTAKA

[1] K. Piwakowski, "CRYPTO4 – The Bytelandian Cryptographer (Act IV)," 2004. [Online]. Avaiable : http:// spoj.com/problems/CRYPTO4/
[2] "Kasiski Method." [Online]. Available: http://pages.mtu.edu/ ~shene/NSF-4/Tutorial/VIG/Vig-Kasiski.html.
[3] K. Devlin, The Joy of Sets: Fundamentals of Contemporary Set Theory, 2nd ed. New York: Springer, 1993.
[4] john_jones, "SPOJ Discussion Board," 2009. [Online]. Available: http://discuss.spoj.com/t/problemset-3/242/13.