Kriptoanalisis Cipher Substitusi Alfabetik dengan Algoritma Genetis

Cil Hardianto Satriawan / 13508061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18061@students.if.itb.ac.id

Abstrak---Dilakukan kriptoanalisis pada *cipher* berbasis substitusi monoalfabetik dengan panjang kunci yang ditentukan menggunakan algoritma genetik yang diajukan. Dilakukan pembangkitan acak kunci-kunci awal dan dilakukan perhitungan kecocokan antara *cipher text* yang telah di-*decrypt* dengan calon kunci terhadap suatu korpus atau teks referensi yang frekeunsi kemunculan huruf dan kumpulan hurufnya dianggap representatif. Kunci yang terpilih dimutasi silang dan hasilnya diseleksi ulang secara iterative sampai didapatkan tingkat kecocokan yang sesuai. Hasil dibandingkan dengan metode pemecahan secara *brute force*. Diajukan beberapa arah pengembangan lebih lanjut untuk mendapatkan hasil yang lebih baik.

Kata kunci: *cipher substitusi monoalfabetik, algoritma genetic*

1. Pendahuluan

Dasar dari penggunaan enkripsi yang menggunakan kunci tersebar adalah asumsi bahwa terdapat jumlah kemungkinan kunci yang sangat banyak sehingga pengunaan metode *brute force* tidak memungkinkan atau tidak mudah untuk dilakukan dalam waktu singkat. Namun, asumsi dasar ini seringkali tidak benar, karena dalam kunci yang digunakan dan metode enkripsi yang digunakan terdapat pola-pola tertentu yang dapat dimanfaatkan untuk memperoleh kunci secara lebih efisien daripada metode *brute force.*

Ini terutama berlaku pada kasus enkripsi menggunakan metode substitusi huruf, disebut sebagai substitusi mono alfabetik atau secara klasik disebut sebagai Vigenere Cipher. Makalah ini bertujuan menunjukkan kelemahan metode enkripsi tersebut dan menjelaskan serta menunjukkan pemecahan melalui penggunaan algoritma genetic.

2. Vigenere Cipher dan Cipher Subtitusi Alfabetik

Cipher substitusi huruf atau substitusi alfabetik merupakan salah satu sandi awal yang dikembangkan dalam sejarah. Pada metode enkripsi ini, setiap huruf dalam *plain text* ditukar atau disubstitusi dengan huruf lain untuk mendapatkan *cipher text*. Sebagai contohnya, dalam implementasi sederhana, satu huruf ditetapkan sebagai kunci, dengan semua karakter pada *plain text* digeser sebanding dengan nilai angka dari huruf tersebut untuk memperoleh *cipher text*. Apabila terdapat *plain text* “foobar”, maka enkripsi substitusi sederhana menggunakan huruf ”m” sebagai kunci akan menghasilkan *cipher text*:

$$M(x) = PAANAD$$

dimana

$x = (F, O, O, B, A, R)$

Untuk kasus Vigenere Cipher, konsep metode penukaran sederhana dikembangkan untuk mendapatkan penukaran yang berbeda nilai untuk setiap huruf pada *plain text* sepanjang kunci yang diberikan. Sebagai contohnya, *plain text* "foobar" apabila dienkripsi menggunakan kunci "raboof" akan menghasilkan *cipher text* "toppot".

3. Karakteristik Cipher Substitusi dan Kriptoanalisis

Dengan anggapan bahwa yang dipertimbangkan hanya huruf alphabet saja, tanpa spasi, tanda baca, maupun karakter khusus, maka terdapat sejumlah 26^n kemungkinan kunci dalam dekripsi sepotong *plain text*, di mana *n* merupakan panjang kunci yang digunakan untuk melakukan enkripsi. Apabila metode *brute force* digunakan, maka percobaan semua kombinasi akan memakan waktu yang lebih lama dari umur alam semesta menggunakan teknologi modern, dengan panjang kunci yang pendek (sekitar 10 karakter). Namun, ada beberapa karakteristik dari *cipher* ini yang memungkinkan seorang pemecah sandi untuk memperoleh hasil yang lebih efisien.

Karakteristik pertama adalah distribusi frekeunsi penggunaan huruf dalam bahasa yang tidak merata. Dalam mempertimbangkan frekeunsi penggunaan huruf, sering digunakan model *n-gram*, di mana *unigram* adalah 1 huruf, *bigram* pasangan huruf, dan *trigram* adalah 3 huruf. Sebagai contohnya, dalam bahasa Inggris, *unigram* yang paling sering muncul (frekeunsi penggunaan tertinggi) adalah *E*, *bigram* yang paling sering digunakan adalah *TH*, dan *trigram* yang paling sering digunakan adalah *THE*. Melalui analisis teks dari korpus sastra bahasa Inggris [1], didapatkan frekuensi kemunculan unigram sebagai berikut:

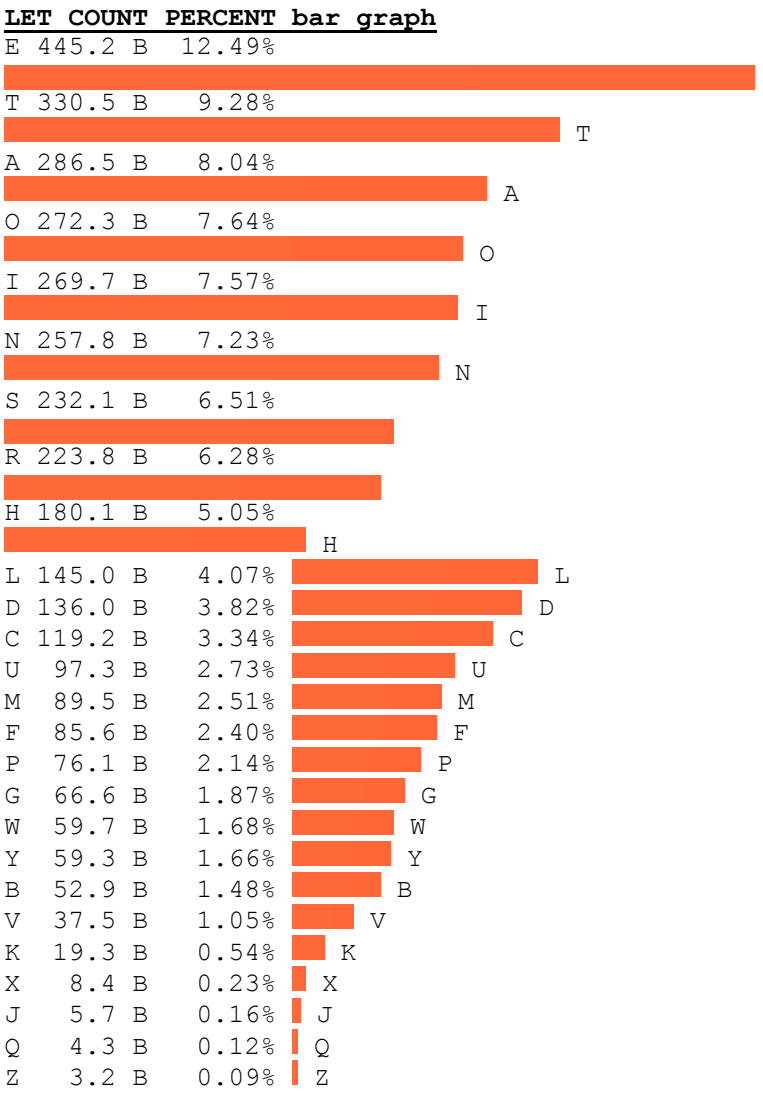

fig 1: Frekuensi unigram dalam bahasa Inggris

Untuk gram dengan nilai *n* yang lebih tinggi, ukuran matriks kemungkinan yang dihasilkan meningkat secara eksponensial, dengan persamaan 26^n. Secara umum, melakukan analisis dengan model ini dengan nilai *n* di atas 3 kurang memungkinkan.

Karakteristik kedua adalah bahwa dalam pemecahan sandi substitusi alfabetik, terdapat kemungkinan bahwa *satu kunci lebih mendekati jawaban yang benar daripada kunci lain*. Ini sangat penting dalam penerapan algoritma genetik, dan terutama dapat terlihat pada *plain text* yang berukuran besar dan sangat besar. Sebagai contoh, apabila terdapat *cipher text :*

TTLERUECXLGML

penggunaan kunci *TELAR* akan menghasilkan dekripsi yang lebih baik daripada *ABCDE*:

TELAR -> APAEABARXUNIA

ABCDE -> TSJBMUCATGGK

Apabila *plain text* yang diharapkan adalah *APAKABARDUNIA*, maka secara objektof kunci yang pertama digunakan lebih baik daripada kunci yang kedua, dan apabila dilihat oleh manusia, terdapat kemungkinan bahwa isi pesan yang dienrkipsi dapat tertebak.

Kedua karakteristik ini dapat kemudian dimanfaatkan untuk mengembangkan algoritma genetis untuk melakukan pemecahan sandi.

4. Algoritma Genetis

Algoritma genetis merupakan pengembangan dari algoritma lebih mendasar yang disebut algoritma evolusioner[2]. Algoritma eviousioner ini mengambil pendekatan *metahueristic* terhadap pemecahan masalah. Dalam pendekatan ini, pengacakan tertentu dilakukan untuk mendapatkan pemecahan masalah yang lebih optimum untuk masalah yang bersifat rumit secara komputasi tetapi tidak rumit untuk diidentifikasi oleh manusia. Contoh dari tipe masalah seperti ini adalah pemecahan *cipher text* yang ditunjukkan di atas, di mana mungkin sulit bagi computer untuk menebak isi pesan dari beberapa huruf saja, namun relatif mudah bagi manusia.

Secara garis besar, algoritma ini bekerja sebagai berikut: bangkitkan secara acak populasi calon jawaban permasalahan, nilai setiap calon jawaban tersebut, ambil jawaban-jawaban yang dinilai paling tinggi, acak parameter jawaban tersebut (mutasi), dan ulangi proses tersebut sampai kualitas jawaban mencapai tingkat yang diinginkan, umumnya sampai pada tingkat di mana jawaban tersebut dapat secara mudah dikenali oleh manusia.

Pada algoritma genetis, dilakukan modifikasi pada proses pembangkitan calon jawaban untuk iterasi berikutnya. Apabila pada algoritma evolusioner dilakukan pengacakan calon jawaban secara individu, pada algoritma genetis dilakukan mutasi silang antara pasangan jawaban yang dinilai paling tinggi pada tiap generasinya.

Untuk dapat melakukan penilaian dan seleksi calon jawaban pada setiap iterasinya, diperlukan metode perthitungan kecocokan calon jawaban terhadap jawaban sebenarnya. Karena pada kenyataannya justru jawaban tersebut yang dicari, diperlukan jawaban yang dianggap representatif. Dalam kasus pemecahan sandi bahasa, maka dilakukan analisis frekeunsi *n-gram* terhadap bahasa tersebut untuk memperoleh model bahasa, yang kemudian dibandingkan dengan frekeunsi *n-gram* yang didapatkan dari pemecahan *cipher text* dengan calon kunci. Fungsi kecocokan yang dihasilkan berbentuk:

$f(K_c) = sum(R^u_i - P^u_i) + sum(R^b_{ij} - P^b_{ij}) + sum(R^t_{ijk} - P^t_{ijk})$

di mana

K_c adalah Kunci calon,

$R^u_i - P^u_i$ adalah pengurangan semua unigram

$R^b_{ij} - P^b_{ij}$ pengurangan semua bigram, dan

$R^t_{ijk} - P^t_{ijk}$ pengurangan semua trigram

Dalam kasus ini, karena dilakukan *pengurangan* antara frekuensi *n-gram* referensi terhadap *n-gram* calon jawaban, maka dicari $f(K_c)$ yang paling kecil. Iterasi algoritma genetis dihentikan ketika fungsi mencapai titik kecil tertentu.

5. Implementasi

Aplikasi yang dihasilkan diimplementasikan secara *object-oriented*, dengan fungsi-fungsi utama tercantum di file header berikut:

```
//---------------------------------
-----------
// File name: Genetic.h
// Author: cilsat
//---------------------------------
-----------

#ifndef __Genetic__
#define __Genetic__

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include <fstream>
#include "math.h"

#include "VigenereCipher.h"
```

```
struct FreqGram
{
	float ug[26],
	float bg[26][26],
	float tg[26][26][26];
};

class Genetic
{
public:
	Genetic();
	~Genetic();

	string* getKeys();
	int getGeneration();

	void initSeed();		//
initializes 10 random keys
	void seed();
	// takes two keys and seeds 10
new keys, overwriting the old ones
	bool select();
	// selects two fittest keys
out of seeds, returns false if
required fitness is achieved, else
returns true

	void calcRef(string);
	void calcCan(string);	//
scans cipher text with given key.
analyses frequency of uni, bi, and
trigrams and outputs to struct.

	// calls calcFit to compare
with reference table.
	void calcFit();

private:
	string *keys;
	string parents[2];
	int generation;

	FreqGram candidate;
	FreqGram reference;
};

#endif
```

Dilakukan pembacaan dari *corpus* (teks referensi) yang diambil dari *e-book* open source, yakni movel *Jane Austen* berjudul *Pride and Prejudice*. Analisis frekeunsi dilakukan dengan membaca file teks, dengan spasi, tanda baca, dan karakter khusus diabaikan. Untuk setiap karakter, dilakukan juga penambahan *bigram* dan *trigram* untuk dua huruf dan tiga huruf di depannya. Perhitungan ini dimasukkan ke dalam struktur penyimpanan khusus, yang menyediakan *static array* untuk unigram (26 karakter), bigram (676 karakter), dan trigram (17576 karakter) yang didapatkan dari teks. Untuk setiap unigram (misal huruf *E*), bigram (*HE*), dan trigram (*THE*), dilakukan perhitungan *running average* untuk memperoleh frekuensi kemunculan uni/bi/trigram tersebut terhadap total keseluruhan. Tabel ini dibangun terpisah dari program utama dikarenakan *load*nya yang cukup besar.

Pengguna memasukkan *cipher text* yang ingin didekripsi dengan mencantumkan nama file, kemudian diminta memasukkan nilai maksimal yang diharapkan dapat dicapai. Ini merupakan subtraksi dari fungsi kecocokan yang didapatkan di subbab sebelumnya. Jumlah "anak" per generasi yang ingin dihasilkan juga dimasukkan.

Ketika program dijalankan, dilakukan pembangkitan acak kunci sejumlah "anak" yang dispesifikasikan. Untuk setiap calon kunci ini, dilakukan dekripsi terhadap *cipher text* masukan menggunakan kunci, kemudian dilakukan analisis frekeunsi yang serupa dengan yang dilakukan kepada teks referensi. Kedua tabel frekuensi n-gram yang dihasilkan kemudian dihitung menggunakan fungsi kecocokan dan diperoleh suatu nilai.

Dua calon kunci yang dinilai paling tinggi kemudian dimutasi silang dengan proses sebagai berikut:

1. Ambil secara langsung 1/2 dari huruf kedua orang tuanya pada posisi acak, sehingga didapatkan kunci baru yang merupakan gabungan acak kedua orang tuanya.
2. Lakukan pengacakan pada setengah dari hurufnya (pada posisi acak) sejumlah maksimal 5 posisi huruf.

Sebagai contoh, apabila panjang kunci adalah 4, maka calon kunci *TEAR* dan calon kunci *ASDF* dapat menghasilkan anak:

1. *T __R + _SD_ -> TSDR*
2. *T_D_ -> TUDP*

Setelah jumlah anak yang dihasilkan sudah tepat, proses diulangi kembali, mulai dari pembuatan tabel frekuensi n-gram calon jawaban.

5. Hasil

Hasil yang didapatkan dari metode ini terbilang cukup mengecewakan untuk kasus di mana *cipher text* pendek, namun cukup baik ketika *cipher text* cukup panjang, terutama apabila dibandingkan dengan metode *brute force* biasa. Dalam kasus ini, *cipher text* yang digunakan berupa satu bab penuh dari novel berbahasa Inggris, yakni 1381 huruf.

Dengan masukan program jumlah anak 10, fungsi kecocokan yang diharapkan 0.91, dan panjang kunci yang digunakan 5 karakter, dan dengan jumlah generasi dibatasi hingga 100, kunci yang memenuhi target fungsi kecocokan berhasil ditemukan pada 3 percobaan dari 10 kali percobaan total.

Apabila dibandingkan dengan hasil yang didapatkan secara *brute force*, maka hasil di atas dapat diterima, namun apabila dibandingkan dengan hasil yang didapatkan dari implementasi *metaheuristic* lain yang lebih modern seperti *particle swarm optimization*, hasil ini terbilang masih kurang. Bahkan, ketika dibandingkan dengan implementasi *evolutionary algorithm* lain, hasil ini cukup mengecewakan.

6. Analisis

Secara keseluruhan, masih terdapa banyak kekurangan dalam implementasi program. Pada beberapa kejadian, program masih terdapat *bug* yang mengakibatkan terjadinya *stack overflow* atau *end-of-buffer error*. Namun, terlepas dari kesalahan teknis dalam implementasi, masih terdapat beberapa bagian model yang dapat dikembangkan lagi untuk mendapatkan hasil yang lebih baik.

Cara paling mudah untuk melakukan perbaikan adalah dengan memodifikasi fungsi kecocokan yang digunakan untuk memperhitungkan *weighting* yang berbeda antara unigram, bigram, dan trigram.

Cara lain adalah dengan menerapkan fungsi kecocokan yang lebih modern seperti yang terdapat pada *particle swarm optimization* atau *simulated annealing implementation*, yang berada di luar lingkup dari makalah ini.

7. Kesimpulan

Penggunaan algoritma genetis dalam pemecahan sandi berbasis substitusi alfabetik merupakan salah satu metode yang terbilang cukup menjanjikan, meskipun terhambat oleh impkementasi yang terbilang rumit dan teori dasar yang juga cukup sulit untuk dimengerti. Diharapkan pada implementasi-implementasi mendatang diterapkan metode-metode yang lebih modern sehingga hasil yang diperoleh dapat lebih baik lagi.

References

[1] M.S. Mayzner, *Tables of Single-letter and Diagram Frequency Countsfor Various Word-length and Letter-position Combinations*, Psychonomic Press, 1965.

[2] A. Clark & E. Dawson, *A parallel genetic algorithm for cryptoanalysis of the polyalphabetic substitution cipher*, Cryptologia 21 no. 2.

[3] John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*, The University of Michigan Press, 1975.

[4] R. Hilton, *Automated cryptanalysis of monoalphabetic substitution ciphers using stochastic optimization algorithms*, University of Colorado, 2012.

[5] I. Rechenberg, *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment Translation no. 1122, B.F. Toms, Trans., Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants, 1965.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

ttd

Nama dan NIM