Beberapa Aplikasi Kriptografi pada Perang Dunia I

Wafdan Musa Nursakti – NIM: 13507065
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB
Jalan Ganeca no. 10 Bandung
e-mail: zorvanion@gmail.com

Abstrak *– Kerahasiaan informasi sangat penting dalam situasi peperangan. Kriptografi digunakan untuk menyembunyikan informasi yang akan dikirim agar tidak diketahui pihak musuh dengan cara mengubah plainteks (pesan asli) menjadi cipherteks (pesan tersandi). Makalah ini membahas beberapa aplikasi kriptografi dalam Perang Dunia I, yaitu Playfair cipher, Vigenère cipher, dan ADFGVX cipher.*

Kata Kunci*: Kriptografi, Playfair cipher, Vigenère cipher, dan ADFGVX cipher.*

1. PENDAHULUAN

Kriptografi telah dipelajari manusia sejak tahun 400 SM, tepatnya pada zaman Yunani Kuno. Tercatat bahwa kriptografi transposisi merupakan sistem kriptografi yang pertama kali dipergunakan. Dikatakan transposisi karena bentuk penyandiannya mempergunakan metode penukaran posisi abjad (yang masih sederhana). Pada saat itu, untuk memecahkan cipherteks, bagi orang yang tidak mempunyai kuncinya, membutuhkan waktu yang sangat lama. Seiring dengan teknologi yang makin berkembang, cipherteks tersebut (pada masa Yunani Kuno) hanya membutuhkan waktu beberapa detik saja untuk dipecahkan sandinya dengan bantuan komputer, yang memiliki daya komputasi dan pemecahan masalah kompleks yang sangat cepat.

Pada Perang Dunia I, cipher (algoritma kriptografi) transposisi masih populer dengan berbagai variasi dan kombinasi dengan berbagai cipher lainnya. Keandalan suatu cipher berperan penting dalam menentukan kemenangan-kekalahan dalam peperangan. Jika algoritma yang kita gunakan bisa dipecahkan oleh musuh, peluang kekalahan akan menjadi besar.

2. PLAYFAIR CIPHER

2.1. Sejarah Singkat

Playfair cipher atau disebut juga Playfair square ditemukan oleh Sir Charles Wheatstone pada tahun 1854. Meskipun penemunya adalah Wheatstone, nama Playfair cipher diambil dari nama Lord Playfair yang mempromosikan cipher ini secara besar-besaran.

Gambar 1 : Sir Charles Wheatstone dan Lord Playfair

Playfair cipher digunakan oleh tentara Inggris dalam Perang Boer Kedua dan dalam Perang Dunia I. Cipher ini digunakan karena cepat dan tidak membutuhkan peralatan tambahan untuk mengoperasikannya.

Playfair cipher sudah tidak digunakan lagi dalam militer semenjak ditemukannya alat enkripsi digital. Karena cipher ini mudah sekali dipecahkan oleh alat digital modern hanya dalam beberapa detik.

2.2. Cara Kerja

Playfair merupakan digraph cipher, artinya proses enkripsi dilakukan setiap dua huruf. Misalkan plainteksnya "MATEMATIKA", maka menjadi "MA TE MA TI KA". Playfair menggunakan tabel kunci berukuran 5x5 yang berisi kata atau frase kunci.

Untuk membuat tabel kunci, pertama-tama isi ruang kosong pada tabel dengan huruf-huruf dari kata kunci dengan catatan tidak ada huruf yang berulang. Misalkan kita gunakan kata kunci "MISALKAN".

M	I	S	A	L
K	N			

Selanjutnya isi ruang kosong sisanya dengan huruf-huruf alfabet secara berurutan. Huruf "I" dan "J" ditempatkan di ruang yang sama.

M	I	S	A	L
K	N	B	C	D
E	F	G	H	O
P	Q	R	T	U
V	W	X	Y	Z

Selanjutnya plainteks ,dengan spasi diabaikan, yang akan dienkripsi dipecah dua-dua. Misalkan "SERBU MARKAS MUSUH" menjadi "SE RB UM AR KA SM US UH".

Ada empat aturan dalam enkripsi Playfair cipher:

- Jika pasangan huruf berisi huruf yang sama atau huruf terakhir tidak mempunyai pasangan, tambahkan X setelah huruf pertama. Misalkan TREEBEARD menjadi TR E**X** BE AR D**X**.

- Jika pasangan huruf ada pada satu baris dalam tabel kunci, maka huruf pertama diganti menjadi huruf yang ada tepat di kanannya dalam baris itu, demikian juga pasangannya, secara berurutan. Jika huruf ada di bagian paling kanan kolom, maka huruf itu diganti menjadi huruf di paling kiri baris (putar balik). Pada contoh di atas , SM menjadi AI.

M	I	S	A	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

- Jika pasangan huruf ada pada satu kolom yang sama, maka maka huruf pertama menjadi huruf yang ada tepat di bawahnya dalam kolom itu, demikian juga pasangannya, secara berurutan. Jika huruf ada di bagian paling bawah kolom, maka huruf itu diganti menjadi huruf di paling atas kolom (putar balik). Pada contoh di atas , RB menjadi XG.

.	.	.	.	.
.	.	B	.	.
.	.	G	.	.
.	.	R	.	.
.	.	X	.	.

- Jika pasangan huruf tidak berada di kolom maupun baris yang sama, maka ganti dengan pasangan huruf yang membentuk sudut-sudut segi-empat (*square*) dengan keduanya. Urutan huruf pasangan hasil enkripsi sesuai dengan urutan huruf plainteksnya. Pada contoh di atas , US menjadi RL.

.	.	S	.	L
.	.	.	.	.
.	.	.	.	.
.	.	R	.	U
.	.	.	.	.

Sehingga, "SERBU MARKAS MUSUH" menjadi "MGXGPLSTCMAIRLTO".

Proses dekripsi merupakan kebalikan dari proses enkripsi. Yaitu dengan membalik aturan yang empat tadi dengan membuang huruf X tambahan (jika ada).

3. VIGENÈRE CIPHER

3.1. Sejarah Singkat

Dokumen lengkap tentang cipher ini sebenarnya sudah dirumuskan oleh Leon Battista Alberti pada sekitar 1467. Namun pada abad ke-19, cipher ini dipopulerkan kembali oleh Blaise de Vigenère, sehingga nama cipher ini diambil dari namanya.

Gambar 2 : Blaise de Vigenère

Dalam Perang Dunia I, cipher ini digunakan oleh tentara Rusia dalam versi yang sangat kompleks. Namun berhasil dipecahkan oleh kriptanalis Austro-Hungaria, Hermann Pokorny.

3.2. Cara Kerja

Vigenère cipher adalah metode enkripsi teks alfabet yang menggunakan deretan *Caesar cipher* yang berbeda-beda berdasarkan huruf-huruf pada kata kunci.

Dalam *Caesar cipher*, tiap huruf alfabet digeser sepanjang beberapa tempat. Misalkan dalam *Caesar cipher* dengan tiga pergeseran, A menjadi D, B menjadi E, dan seterusnya. Pada Vigenère, digunakan deretan *Caesar cipher* yang berurutan dengan jumlah pergeseran yang berbeda-beda.

Untuk mengenkripsi, digunakan tabel alfabet yang disebut *tabula recta*, atau disebut juga *tabel Vigenère*. Tabel ini terdiri dari semua huruf alfabet yang ditulis 26 kali dalam 26 baris. Tiap baris berisi tabel alfabet di baris atasnya yang hurufnya digeser sekali ke kiri secara siklus.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Gambar 3 : "tabula recta"

Sebagai contoh kasus, plainteks yang akan dienkripsi adalah

SERANGTENGAHHARI

Misalkan kata kunci yang akan dipilih adalah

MELATI

Langkah pertama yang dilakukan adalah mengulang kata kunci sampai sama panjang dengan plainteksnya.

MELATIMELATIMELA

Huruf pertama plainteks adalah S, di enkripsi dengan huruf di baris M, yang merupakan huruf pertama dari kata kunci. Hasilnya adalah huruf yang berada pada baris M dan kolom S, yaitu E. Selanjutnya dilakukan hal yang sama untuk seluruh plainteks.
Sehingga diperoleh:

Plainteks : SERANGTENGAHHARI
Kata Kunci : MELATIMELATIMELA
Cipherteks : EICAGOFIYGTPTECI

Proses dekripsi dilakukan dengan menemukan posisi huruf cipherteks pada baris tabel, lalu ambil label dari kolom sebagai plainteks. Contoh : pada baris M, cipherteks E terdapat pada kolom S, sehingga S diambil sebagai huruf plainteks pertama. Begitu pula selanjutnya.

Secara aljabar, proses enkripsi Vigenère ditulis sebagai

$$C_i \equiv P_i + K_i \pmod{26}$$

Dan proses dekripsinya

$$P_i \equiv C_i - K_i \pmod{26}$$

3.3. Pemecahan Vigenère cipher dengan pengujian Kasiski

Pengujian ini memanfaatkan kenyataan bahwa beberapa kata tertentu yang berulang kemungkinan terenkripsi menggunakan huruf-huruf kunci yang sama, sehingga cipherteksnya juga mempunyai bagian berulang yang bersesuaian dengan bagian berulang pada plainteksnya. Contoh:
Plainteks yang dienkripsi dengan kata kunci ABCDEF (panjangnya 6 karakter) tidak mengenkripsi "CRYPTO" menjadi cipherteks yang sama jika berulang.

Kunci : ABCDEF AB CDEFA BCD EFABCDEFABCD
Plainteks : **CRYPTO** IS SHORT FOR **CRYPTO**GRAPHY
Cipherteks : **CSASXT** IT UKSWT GQU **GWYQVR**KWAQJB

Cipherteks di atas tidak mempunyai kelompok huruf berulang yang bersesuaian dengan plainteksnya. Namun jika kata kunci diganti menjadi ABCD (panjangnya 4 karakter),

Kunci : ABCDAB CD ABCDA BCD ABCDABCDABCD
Plainteks : **CRYPTO** IS SHORT FOR **CRYPTO**GRAPHY
Cipherteks : **CSASTP** KV SIQUT GQU **CSASTP**IUAQJB

Maka ditemukan cipherteks-berulang yang bersesuaian dengan plainteks-berulang, yaitu **CSASTP** bersesuaian dengan **CRYPTO,** dan setiap CRYPTO jika bertemu dengan kunci ABCDAB pasti akan dienkripsi menjadi CSASTP.

Plainteks yang lebih panjang akan membuat pengujian lebih akurat karena biasanya mengandung lebih banyak cipherteks-berulang. Contoh :
Cipherteks:
DYDUXRMHTVDV**NQD**QNW**DYDUXRMH**ARTJGW**NQD**

Jarak antara **DYDUXRMH** yang berulang adalah 18 karakter. Jika kita berasumsi bahwa cipherteks-berulang merepresentasikan plainteks-berulang, maka panjang kata kunci yang mungkin adalah 18, 9, 6, 3, atau 2 karakter (faktor-faktor dari 18).

Sedangkan jarak antara **NQD** yang berulang adalah 20 karakter. Artinya panjang kata kunci yang mungkin adalah 20, 10, 5, 4, atau 2 (faktor-faktor dari 20).

Irisan dari kedua himpunan faktor di atas menghasilkan 2, yang merupakan panjang kata kunci.

Dengan panjang kata kunci diketahui, akan lebih mudah menemukan kata kunci sebenarnya dengan *brute force*, maupun analisis frekuensi (tidak dibahas di sini).

Berikut ini contoh implementasi Vigenere cipher dalam bahasa C++ [7]:

```
//********************************
******

// Name: vigenere-crypt
// Description:encrypt/decrypt
ascii text files with the Vigenere
algorithm. The encryption key is
user determined so that possessing
the program does not automatically
allow one to crack the code.
// By: Mark Rothfuss
//
//
// Inputs:When the program is run
you will be prompted for the name
of the textfile to encrypt or
decrypt, what operation to
perform(encryption or decryption)
and the key with which you wish to
perform the operation with.
//
// Returns:The program generates a
new text file of the encrypted or
decrypted text with a filname of
the form: encrypted-filename.txt
//
//Assumes:The code, as is, is
ready for compilation and use.
Care has been taken to eliminate
most user input errors.
//
//Side Effects:Lines of text
should not be more than 10,000
chars long to prevent memory
overruns. This should not be a
problem for most, however, as
<returns> wihtin the text signify
the start of a new line.
//This code is copyrighted and has
limited warranties.
//Please see http://www.Planet-
Source-
Code.com/xq/ASP/txtCodeId.1793/lng
WId.3/qx/vb/scripts/ShowCode.htm
//for details.
//********************************
******

//--------------------------------
--------------------------------
----
// Written by: Mark Rothfuss
// Last modified: 5/20/01
//
// Description: Uses the Vigenere
encryption algorithm to
// encrypt/decrypt ascii text
within text files. The
// encryption key is user
definable and may be of any
// length.
//
//--------------------------------
--------------------------------
----
#include <iostream.h>
#include <fstream.h>
#include <string.h>
char* encrypt(char*, char*);
char* decrypt(char*, char*);
int keyPos=0;
int const lineLength=10000;
//--------------------------------
--------------------------------
----


    void main(int argc, char
**argv) {
    char *inputfile;


        if (argc==1) {
        cout << "Enter name of
file to encrypt/decrypt: ";
        char temp[lineLength];
        cin >> temp;
        inputfile = new
char[strlen(temp)];
        inputfile=temp;
    }
    ifstream in;
    in.open(inputfile);


        if (in==NULL) {
        cerr << "Error, could not
open the file " << inputfile <<
endl;
        exit(0);
    }
    char crypt[lineLength]="";


        while
(strcmp(crypt,"E")!=0 &&
```

```
strcmp(crypt,"D")!=0) {
        cout << "[E]ncrypt or
[D]ecrypt? ";
        cin >> crypt;
strupr(crypt);
    };
    char key[lineLength];
    cout << "Enter key: ";
    cin >> key;
    char *outputfile;
    if (strcmp(crypt,"E")==0)
outputfile=strcat("Encrypted-",
inputfile);
    if (strcmp(crypt,"D")==0)
outputfile=strcat("Decrypted-",
inputfile);
    ofstream out;
    out.open(outputfile);


        if (out==NULL) {
        cerr << "Error, could not
open the file " << argv[1] <<
endl;
        exit(0);
    }
    char line[lineLength];
    char *cryptedline;


        while
(in.getline(line,lineLength-1)) {
        if (strcmp(crypt,"E")==0)
cryptedline=encrypt(key, line);
        if (strcmp(crypt,"D")==0)
cryptedline=decrypt(key, line);
        out.write(cryptedline,
strlen(cryptedline));
        out.put('\n');
    }
    return 0;
}
//--------------------------------
--------------------------------
----


    char *encrypt(char *key, char
*line) {
    char *temp = new
char[lineLength];
    int i,lenKey=strlen(key);


        for (i=0 ; i<=strlen(line)
; i++) {


            if (line[i]>=32 &&
line[i]<=126) {
            temp[i] = 32 +
((line[i]-32) + (key[keyPos]-
32))%95;
            keyPos++;
            if (keyPos >= lenKey)
keyPos=0;
        }
        else temp[i] = line[i];
    }
    temp[i+1]='\0';
    return temp;
}
//--------------------------------
--------------------------------
----


    char *decrypt(char *key, char
*line) {
    char *temp = new
char[lineLength];
    int i,lenKey=strlen(key);


        for (i=0 ; i<=strlen(line)
; i++) {


            if (line[i]>=32 &&
line[i]<=126) {
            temp[i] = 32 + line[i]
- key[keyPos];
            if (temp[i]<32)
temp[i] += 95;
            keyPos++;
            if (keyPos >= lenKey)
keyPos=0;
        }
        else temp[i] = line[i];
    }
    temp[i+1]='\0';
    return temp;
}
//--------------------------------
--------------------------------
----
```

4. ADFGVX CIPHER

4.1. Sejarah Singkat

ADFGVX cipher adalah cipher yang digunakan oleh tentara Jerman dalam Perang Dunia I. ADFGVX sebenarnya merupakan perluasan dari ADFGX cipher. ADFGX ditemukan oleh Kolonel Fritz Nebel. Cipher ini dinamai demikian berdasarkan huruf-huruf yang mungkin ada pada cipherteksnya. Alasan dipilihnya huruf ADFGX adalah karena bila huruf-huruf ini disampaikan lewat telegram, beda bunyinya terdengar jelas, sehingga meminimalkan terjadinya kesalahan operator.

ADFGVX cipher berhasil dipecahkan oleh Letnan Georges Painvin dari Perancis pada tahun1918 beberapa minggu setelah Jerman menjalankan Penyerangan Musim Panas (*Spring Offensive*). Hal ini menyebabkan tentara Perancis mampu membaca gerakan penyerangan Jerman itu dan pada akhirnya berhasil menghentikannya.

4.2. Cara Kerja

ADFGX cipher termasuk penyandian transposisi yang dikombinasikan dengan *Polybius square* dengan transposisi kolumnar.

Misalkan kita mempunyai plainteks "SERBUSEGERA". Langkah pertama, campuran huruf alfabet rahasia diisikan pada *Polybius square* 5x5. Di dalamnya huruf I dan J disatukan.

	A	**D**	**F**	**G**	**X**
A	A	G	L	O	Q
D	R	B	H	M	P
F	V	S	C	I	N
G	Y	W	T	D	K
X	F	Z	X	U	E

Dengan menggunakan *square* di atas, plainteks diubah menjadi bentuk terfraksi :

S E R B U S E G E R A
FD XX DA DD XG FD XX AD XX DA AA

Langkah kedua, plainteks yang terfraksi ditulis dalam sebuah "tabel" secara baris di bawah sebuah kata kunci transposisi, misalkan "MOBIL".

M	**O**	**B**	**I**	**L**
F	D	X	X	D
A	D	D	X	G
F	D	X	X	A
D	X	X	D	A
A	A			

Langkah ketiga, urutkan huruf pada kata kunci transposisi secara alfabet, huruf-huruf yang ada di kolom bawahnya tetap mengikuti.

B	**I**	**L**	**M**	**O**
X	X	D	F	D
D	X	G	A	D
X	X	A	F	D
X	D	A	D	X
			A	A

Langkah terakhir, "tabel" dibaca secara kolom berurutan dari kiri ke kanan, sehingga cipherteks yang dihasilkannya adalah

XDXX XXXD DGAA FAFDA DDDXA

Dalam praktek di lapangan, kata kunci transposisi sangat panjang. Selain itu, baik kata kunci transposisi maupun tabel alfabet diganti setiap hari.

Pada tahun 1918, cipher ini ditambahi huruf V. Sehingga *Polybius Square* ukurannya menjadi 6x6. Akibatnya, dimungkinkan seluruh huruf alfabet beserta angka 0-9 untuk masuk dalam *Polybius square*.

5. KESIMPULAN

Teknologi yang berkembang menyebabkan algoritma kriptografi juga berkembang. Algoritma-algoritma kriptografi yang dipakai pada Perang Dunia I sudah bisa dipecahkan dengan mudah oleh bantuan komputer modern saat ini.

Di samping beberapa kekurangan dalam aplikasinya, algoritma-algoritma kriptografi pada masa Perang dunia I merupakan inspirasi bagi berbagai algoritma kriptografi pada masa Perang Dunia II hingga masa modern selanjutnya. Karena itu, ini merupakan sejarah yang pantas dipelajari dan diingat.

DAFTAR REFERENSI

[1] Wikipedia
http://en.wikipedia.org/wiki/Kasiski_examination. Tanggal akses : 29 Desember 2008 pukul : 22.00

[2] Wikipedia
http://en.wikipedia.org/wiki/Vigenère_cipher Tanggal akses : 29 Desember 2008 pukul : 21.00

[3] Wikipedia
http://en.wikipedia.org/wiki/ADFGVX_cipher Tanggal akses : 29 Desember 2008 pukul : 20.30

[4] Wikipedia
http://en.wikipedia.org/wiki/Playfair_cipher Tanggal akses : 29 Desember 2008 pukul : 20.00

[5] NOVA Online _ Decoding Nazi Secrets. http://www.pbs.org/wgbh/nova/decoding/playfair2.html

[6] Kriptografi_ Sejarah Perkembangan [2] « Asian Advantage
http://suray.wordpress.com/2007/05/11/kriptografi-sejarah-perkembangan-2/

[7] vigenere-crypt by Mark Rothfuss
https://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=1793&lngWId=3