# Modifikasi *Playfair Cipher* Menggunakan *Vigenere Cipher*

**Meirza Auriq – NIM : 13504103**
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10Bandung,
Email: if14103@students.if.itb.ac.id

***Abstract –*** *Kriptografi atau yang sering dikenal dengan sebutan ilmu penyandian data, dapat dibagi menjadi dua klasifikasi yaitu kriptografi kalsik dan kriptografi modern. Kriptografi klasik merupakan awal dari perkembangan kriptografi yang pada zaman modern ini. Beberapa teknik kriptografi klasik yang populer antara lain Playfair Cipher dan Vigenere Cipher. Playfair cipher merupakan salah satu teknik kriptografi dimana pesan dienkripsi berdasarkan pasangan huruf (bigram), bukan huruf tunggal seperti pada cipher klasik lainnya. Vigenere cipher merupakan salah satu metode peng-enkripsi-an teks alphabet menggunakan seri lain dari Caesar cipher yang berdasarkan suatu kunci. Namun kedua teknik ini masih mempunyai kelemahannya masing-masing. Karena itu, di dalam makalah ini akan dicoba untuk memodifikasi Playfair Cipher dengan Vigenere Cipher dengan membangun sebuah program kriptografi sederhana. Hasil yang dicapai diharapkan dapat meningkatkan keamanan dalam pengiriman pesan sehingga dapat memberikan jaminan integritas data serta menjaga kerahasiaan. Program ini digunakan untuk mengetahui perbandingan antara plainteks awal dengan plainteks yang telah melalui proses enkripsi dan dekripsi program serta tingkat keamanan yang dihasilkan. Plainteks yang telah melalui proses tersebut dapat berbeda dengan plainteks awal karena melalui proses pengaturan dari Playfair Cipher. Sedangkan keamanan untuk cipherteks hasil modifikasi Playfair Cipher dengan Vigenere Cipher sangat tinggi*

***Kata Kunci****: playfair cipher, bigram, vigenere.*

## 1 PENDAHULUAN

### 1.1 Latar Belakang

Kriptografi atau yang sering dikenal dengan sebutan ilmu penyandian data, adalah suatu bidang ilmu dan seni yang bertujuan untuk menjaga kerahasiaan suatu pesan yang berupa data-data dari pihak-pihak lain yang tidak berhak sehingga tidak menimbulkan kerugian. Pada saat ini, ilmu bidang kriptografi dapat dibagi menjadi dua klasifikasi yaitu kriptografi klasik dan kriptografi modern.

Kriptografi klasik sudah mulai dipelajari manusia sejak tahun 400 SM, yaitu pada zaman Yunani kuno. Dari catatan bahwa "Penyandian Transposisi" merupakan sistem kriptografi pertama yang digunakan atau dimanfaatkan. Bidang ilmu ini terus berkembang seiring dengan kemajuan peradaban manusia, dan memegang peranan penting dalam strategi peperangan yang terjadi dalam sejarah manusia, mulai dari sistem kriptografi "*Caesar Chiper*" yang terkenal pada zaman Romawi kuno, "*Playfair Cipher*" yang digunakan Inggris dan "ADFVGX *Cipher*" yang digunakan Jerman pada Perang Dunia pertama, hingga algoritma-algoritma kriptografi rotor yang populer pada Perang Dunia II , seperti Sigaba / M-134 (Amerika Serikat), Typex (Inggris), Purple (Jepang), dan mesin kriptografi legendaris Enigma (Jerman).

Teknik kriptografi klasik dibagi menjadi dua, yaitu dengan teknik transposisi dan substitusi. Pada teknik transposisi, huruf-huruf dalam plainteks hanya dimanipulasi letak posisinya (*transpose*) untuk menjadi teks sandi. Sedangkan pada teknik substitusi, setiap huruf dalam plainteks akan tepat berkorespondensi satu-satu dengan huruf dalam teks sandinya. Untuk penggunaan yang lebih lanjut agar didapat skema yang lebih kompleks, digunakan beberapa teknik improvisasi dari teknik-teknik tersebut.

Sedangkan kriptografi modern merupakan algoritma kriptografi yang berkembang pada zaman modern ini. Kriptografi modern beroperasi dalam bit atau byte, tidak seperti kriptografi klasik yang hanya beroperasi pada karakter. Namun pada prinsipnya metode yang digunakan pada kriptografi modern merupakan metode yang diadopsi kriptografi klasik, namun dengan bantuan komputer dapat dibuat sedemikian rupa sehingga menjadi lebih rumit.

### 1.2 Ruang Lingkup

Untuk tidak memperluas area pembahasan yang terdapat pada makalah, maka penulis memfokuskan permasalahan pada dua algoritma kriptografi klasik, yaitu *Playfair Cipher* dan *Vigenere Cipher*.

### 1.3 Tujuan Penulisan

Tujuan yang hendak dicapai dari penulisan makalah ini adalah untuk memodifikasi *Playfair Cipher* dengan *Vigenere Cipher* dan juga melakukan analisis terhadap keamanannya. Hasil yang dicapai diharapkan dapat meningkatkan keamanan dalam pengiriman pesan sehingga dapat memberikan jaminan integritas data serta menjaga kerahasiaan.

## 2 DASAR TEORI

### *2.1 Playfair Cipher*

*Playfair cipher* merupakan salah satu teknik kriptografi dimana pesan dienkripsi berdasarkan pasangan huruf (bigram), bukan huruf tunggal seperti pada cipher klasik lainnya. *Playfair cipher* menggunakan kunci 25 huruf yang disusun dalam bujursangkar dengan menghilangkan huruf J dari abjad. Susunan kunci di dalam bujursangkar diperluas dengan menambahkan kolom keenam dan baris keenam seperti di bawah ini [1].

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

**Tabel 1. Tabel kunci bujursangkar**

Penambahan susunan kunci pada kolom keenam dan baris keenam dimaksudkan untuk mempermudah dalam melakukan proses penyandian.

Dari tabel kunci tersebut, jumlah kemungkinan kunci tersebut adalah
25! = 15.511.210.043.330.985.984.000.000

Misalkan ada sebuah pesan PROPOSAL MAKALAH KRIPTOGRAFI akan dienkripsi menggunakan playfair cipher maka pesan akan diatur dahulu sebagai berikut:

PR OP OS AL MA KA LA HK RI PT OG RA FI

Pesan yang sudah diatur kemudian dienkripsi menggunakan contoh kunci yang ada di atas dengan algoritma enkripsi sebagai berikut:

1. Jika dua huruf terda pada baris kunci yang sama maka tiap huruf diganti dengan huruf di kanannya.
2. Jika dua huruf terdapat pada kolom kunci yang sama maka tiap huruf diganti dengan huruf di bawahnya.
3. Jika dua huruf tidak pada baris yang sama atau kolom yang sama, maka huruf pertama diganti dengan huruf pada perpotongan baris huruf pertama dengan kolom huruf kedua. Huruf kedua diganti dengan huruf pada titik sudut keempat dari persegi panjang yang dibentuk dari 3 huruf yang digunakan sampai sejauh ini.

Hasil chipertext adalah LI QY SQ YA EF NL AYCS IP WO EL PM.

### 2.2 Vigenere Cipher

*Vigenere cipher* merupakan salah satu metode peng-enkripsi-an teks alphabet menggunakan seri lain dari *Caesar cipher* yang berdasarkan suatu kunci. *Vigenere Cipher* merupakan contoh sederhana dari algoritma enkripsi subtitusi abjad majemuk (*Polyalphabetic Subtitution Cipher*) [4]. *Vigenere Cipher* menggunakan sebuah tabel alphabet yang disebut tabel *vigenere* atau *tabula recta*.

| | 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 |

**Tabel 2. Tabel vigenere atau *tabula recta***

Pada algoritma *Vigenere Cipher*, huruf yang sama pada plainteks tidak selalu menjadi huruf yang sama pula pada cipherteks, tetapi tergantung dari huruf kunci yang digunakan. Vigenere Cipher dapat mencegah frekuensi huruf di dalam cipherteks yang mempunyai pola tertentu yang sama seperti pada cipher abjad tunggal. Akan tetapi, jika periode kunci diketahui dan tidak telalu panjang, maka kunci dapat ditentukan dengan melakukan *exhaustive key search*.
Vigenere juga dapat dilihat dalam bentuk aljabar. Jika huruf A-Z diberi nomor 0-25, dan kemudian dilakukan operasi modulo 26, maka proses enkripsi dapat ditulis dalam persamaan (1),

$$C_i \equiv (P_i + K_i) \bmod 26$$

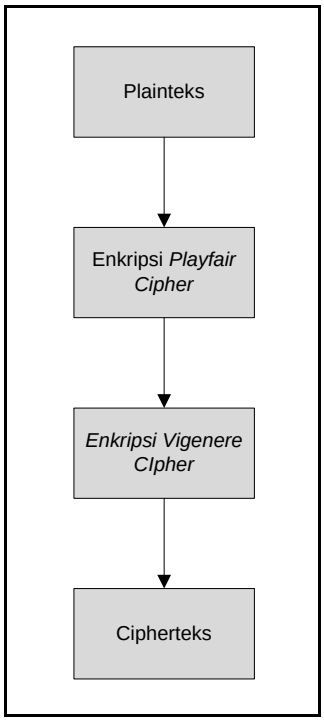
dan proses dekripsi dalam persamaan (2),

$$P_i \equiv (C_i - K_i) \bmod 26$$

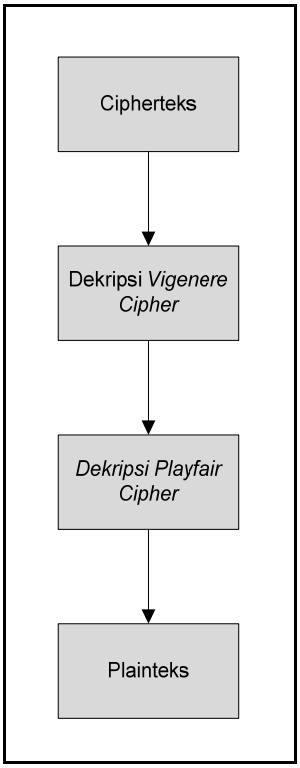
## 3 PERANCANGAN PROGRAM

Dalam pembuatan makalah ini, penulis merancang sebuah program kriptografi sederhana dengan metode *Playfair Cipher* yang dimodifikasi dengan *Vigenere Cipher* untuk mempermudah analisis yang akan dilakukan. Modifikasi yang dilakukan penulis dalam makalah ini adalah dengan menambahkan algoritma *Vigenere Cipher* setelah proses *Playfair Cipher* telah selesai dilakukan. Untuk lebih jelasnya, proses enkripsi dan dekripsi pada *Playfair Cipher* setelah dimodifikasi dengan *Vigenere Cipher* dapat dilihat

pada bagan pada gambar berikut:




**Gambar 1. Bagan enkripsi hasil modifikasi**




**Gambar 2. Bagan dekripsi hasil modifikasi**

Secara teknis, program ini terdiri dari 3 bagian, yaitu:

1. Membuat tabel kunci *Playfair* dari kata kunci
   Tahap ini akan menciptakan sebuah tabel kunci 5x5 yang diperoleh dari kata kunci yang dimasukan.
2. Melakukan proses ekripsi
   Tahap ini akan mengubah plainteks menjadi cipherteks dengan algoritma enkripsi *Playfair Cipher* dan *Vigenere Cipher.*
3. Melakukan proses dekrispi
   Tahap ini akan mengembalikan cipherteks menjadi plainteks seperti semula.

Ketiga tahap tersebut akan diterangkan lebih rinci pada sub bab selanjutnya.

### 3.1 Pengolahan kunci

Kunci yang baik adalah kunci yang dibangkitkan dari kalimat yang mudah diingat sehingga dapat menjadi sebuah solusi dalam melakukan proses penyandian.

Dalam program yang dibuat oleh penulis, kata kunci yang digunakan memiliki dua fungsi. Fungsi kunci yang pertama adalah untuk membentuk tabel kunci bujursangkar *Playfair Cipher*. Fungsi yang kedua adalah sebagai kunci enkripsi atau dekripsi *Vigenere Cipher.*

Misalkan kita mempunyai kunci dari kalimat KRIPTOGRAFI KLASIK. Selanjutnya untuk membuat tabel kunci bujursangkar, program yang penulis rancang akan melakukan beberapa langkah, yaitu:

1. Buang huruf yang berulang, spasi dan huruf J jika ada:

   KRIPTOGAFLS

2. Tambahkan semua huruf yang belum ada secara berurut (kecuali huruf J):

   KRIPTOGAFLSBCDEHMNQUVWXYZ

3. Masukan huruf-huruf tersebut ke dalam tabel kunci bujursangkar:

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

**Tabel 3. Tabel kunci bujursangkar**

Sedangkan untuk fungsi kunci yang kedua, program hanya menghilangkan spasi yang terdapat pada kunci dan langsung digunakan dalam proses enkripsi dan dekripsi *Vigenere Cipher*.

### 3.2 Algoritma enkripsi program

Enkripsi merupakan sebuah proses dimana data teks asli diubah menjadi data teks rahasia sehingga informasi yang terkandung tidak dapat dimengerti

artinya. Algoritma enkripsi yang penulis rancang terdiri dari 3 tahap, yaitu:

1. Mengatur plainteks menjadi bigram
2. Melakukan enkripsi *Playfair Cipher*
3. Melakukan enkripsi *Vigenere Cipher*

### 3.2.1 Mengatur plainteks menjadi bigram

Sebelum melakukan proses enkripsi *Playfair Cipher*, program mengatur terlebih dahulu pesan yang akan dienkripsi (plainteks) sebagai berikut:

1. Semua spasi dan karakter yang bukan alfabet dihilangkan dari plainteks.
2. Semua huruf dijadikan huruf capital.
3. Jika terdapat huruf J pada plainteks maka ganti huruf tersebut dengan huruf I.
4. Pesan yang akan dienkripsi ditulis dalam pasangan huruf (bigram).
5. Jika ada huruf yang sama pada pasangan huruf, maka sisipkan huruf X di tengahnya. Huruf X dipilih karena sangat kecil kemungkinannya terdapat huruf X yang sama dalam bigram. Apabila hal ini terjadi maka akan disisipkan huruf Q.
6. Jika huruf pada plainteks adalah ganjil maka tambahkan huruf X di akhir plainteks atau huruf Q jika huruf terakhir plainteks adalah X.

Misalkan kita mempunyai pesan sebagai berikut:
"Kennedy was assassinated by Lee Harvey Oswald"

Setelah plainteks dilakukan pengaturan dengan langkah-langkah yang di atas, maka akan menjadi:

KE NX NE DY WA SA SX SA SX SI NA TE DB YL EX EH AR VE YO SW AL DX

### 3.2.2 Melakukan enkripsi *Playfair Cipher*

Plainteks yang sudah diatur menjadi bigram kemudian akan dienkripsi dengan algoritma *Playfair Cipher* menggunakan tabel kunci bujursangkar yang sudah dibangkitkan. Algoritma enkripsi untuk setiap bigram adalah sebagai berikut:

1. Jika pasangan huruf terdapat pada baris kunci yang sama maka tiap huruf diganti dengan huruf di kanannya (pada kunci yang telah diperluas). Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - OG menjadi GA
   - SE menjadi BS

2. Jika pasangan huruf terdapat pada kolom kunci yang sama maka tiap huruf diganti dengan huruf di bawahnya (pada kunci yang telah diperluas). Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - RM menjadi GW
   - KV menjadi OK

3. Jika pasangan huruf tidak terdapat pada baris dan kolom yang sama maka huruf pertama diganti dengan huruf pada perpotongan kolom huruf pertama dengan baris huruf kedua, sedangkan huruf kedua diganti dengan huruf pada perpotongan kolom huruf kedua dengan baris huruf pertama. Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - KD menjadi SP
   - EH menjadi US

Jika plainteks "Kennedy was assassinated by Lee Harvey Oswald" sudah diatur dan kemudian dienkripsi dengan algoritma *Playfair Cipher* menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK, maka akan menjadi:

ST XI CU QP GX OC VC KC XC LU EC FZ ZC US IG SZ FV VB FO YC

### 3.2.3 Melakukan enkripsi *Vigenere Cipher*

Vigenere Cipher menggunakan tabel *Vigenere* atau tabula recta untuk melakukan proses enkripsinya. Dalam program yang penulis rangcang, algoritma enkripsi *Vigenere Cipher* menggunakan rumus persamaan (1):

$$C_i \equiv (P_i + K_i) \bmod 26$$

Dan himpunan persamaan (3):

$$\{A,B,C,\ldots,Y,Z\} = \{0,1,2,\ldots,24,25\}$$

Jika panjang kunci lebih pendek daripada plainteks, maka kunci diulang secara periodic.

Contoh:
Kunci : mission
Plainteks : ATTACK AT NIGHT
Kunci : missio nm issio

Untuk melakukan enkripsi gunakan rumus persamaan (1) dimana i merupakan huruf ke-n dalam plainteks. Untuk i = 1,

$$C_1 \equiv (P_1 + K_1) \bmod 26$$
$$= (P'A' + K'm') \bmod 26$$
$$= (0 + 12) \bmod 26$$
$$= 12$$
$$C'M' = 12$$

Maka hasil enkripsi seluruhnya adalah:

Plainteks : ATTACK AT NIGHT
Kunci : missio nm issio
Cipherteks : MBLSKY NF VAYPH

Algoritma enkripsi *Vigenere Cipher* pada program akan diletakan setelah algoritma *Playfair Cipher* sehingga plainteks yang akan dienkripsi menggunakan *Vigenere Cipher* merupakan cipherteks hasil dari enkripsi *Playfair Cipher* dan menggunakan kunci yang sama.

Untuk plainteks "Kennedy was assassinated by Lee Harvey Oswald" yang sudah dienkripsi dengan *Playfair Cipher*, lalu dienkripsi kembali menggunakan algoritma *Vigenere Cipher* menggunakan kunci KRIPTOGRAFI KLASIK, maka akan menjadi:

CKFXVIWGGCWMGCGKFMBKMVZACKHJNUK
QQCQNKOPLFYH

### 3.3 Algoritma dekripsi program

Dekripsi merupakan proses kebalikan dari enkripsi dimana cipherteks dikembalikan lagi menjadi plainteks yang dapat dimengerti informasi yang terkandung di dalamnya. Kunci yang digunakan dalam proses enkripsi diperlukan lagi untuk melakukan proses dekripsi.

Proses dekripsi sangat mirip dengan proses enkripsi den lebih mudah dilakukan. Algoritma dekripsi dalam program yang dibuat oleh penulis terdiri dari 2 tahap:

1. Melakukan dekripsi *Playfair Cipher*
2. Melakukan dekripsi *Vigenere Cipher*

Untuk tahap pertama, dekripsi menggunakan *Vigenere Cipher*, digunakan rumus persamaan (2):

$$P_i \equiv (C_i - K_i) \bmod 26$$

Proses ini menggunakan himpunan yang sama seperti proses enkripsi. Apabila $C_i < K_i$, maka $C_i = C_i + 26$.

Contoh:
Kunci : mission
Cipherteks : MBLSKY NF VAYPH
Kunci : missio nm issio
Plainteks : ATTACK AT NIGHT

Untuk tahap yang kedua, dekripsi menggunakan *Playfair Cipher*, cipherteks dikelompokan terlebih dahulu dalam pasangan huruf (bigram) seperti pada saat enkripsi. Kemudian, terapkan algoritma dekripsi yang merupakan kebalikan dari algoritma enkripsi untuk setiap bigram tersebut.

Algoritma dekripsi *Playfair Cipher* adalah sebagai berikut:

1. Jika pasangan huruf terdapat pada baris kunci yang sama maka tiap huruf diganti dengan huruf di kirinya (pada kunci yang telah diperluas). Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - GA menjadi OG
   - BS menjadi SE
2. Jika pasangan huruf terdapat pada kolom kunci yang sama maka tiap huruf diganti dengan huruf di atasnya (pada kunci yang telah diperluas). Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - GW menjadi RM
   - OK menjadi KV
3. Jika pasangan huruf tidak terdapat pada baris dan kolom yang sama maka huruf pertama diganti dengan huruf pada perpotongan kolom huruf pertama dengan baris huruf kedua, sedangkan huruf kedua diganti dengan huruf pada perpotongan kolom huruf kedua dengan baris huruf pertama. Contoh menggunakan tabel kunci yang dibangkitkan dari kalimat KRIPTOGRAFI KLASIK (tabel 3):
   - KD menjadi SP
   - US menjadi EH

## 4 ANALISIS

### 4.1 Analisis hasil program

Analisis hasil program yang dilakukan disini adalah analisis perbandingan antara plainteks awal dengan plainteks yang telah melalui proses enkripsi dan dekripsi program.

Menggunakan contoh sebelumnya, pesan "Kennedy was assassinated by Lee Harvey Oswald" dienkripsi program menggunakan kunci dari kalimat KRIPTOGRAFI KLASIK. Hasilnya adalah:

CKFXVIWGGCWMGCGKFMBKMVZACKHJNUK
QQCQNKOPLFYH

Cipherteks ini kemudian didekripsi kembali oleh program menggunakan kunci yang sama. Hasilnya adalah:

KE NX NE DY WA SA SX SA SX SI NA TE DB YL
EX EH AR VE YO SW AL DX

Apabila pesan di atas dibaca maka akan terlihat seperti:

"KENXNEDY WAS ASXSASXSINATED BY LEXE HARVEY OSWALDX".

Pesan ini dapat lebih dimengerti apabila huruf X dihilangkan. Huruf X ini muncul akibat pengaturan yang dilakukan sebelum pesan dienkripsi dengan algoritma *Playfair Cipher*. Huruf X tersebut tidak dapat dihapus karena tidak dapat diketahui apakah huruf tersebut merupakan hasil pengaturan atau merupakan bagian dari pesan.

### 4.2 Analisis keamanan cipherteks hasil modifikasi

*Playfair cipher* sangat sulit dipecahkan dengan analisis frekuensi relatif huruf-huruf, namun ia dapat dipecahkan dengan analisis frekuensi pasangan huruf. Akan tetapi analisis frekuensi pasangan huruf membutuhkan 600 kemungkinan digram dibandingkan dengan 26 kemungkinan monogram dan juga membutuhkan banyak cipherteks agar berhasil.

Sedangkan untuk *Vigenere Cipher* lebih mudah untuk dipecahkan dengan adanya metode Kasiski. Metode ini membantu menemukan panjang kunci. Setelah panjang kunci diketahui, maka langkah berikutnya adalah menentukan kata kunci dengan menggunakan *exhaustive key search*. Jika panjang kunci adalah p, maka jumlah kunci yang harus dicoba adalah $26^p$. Namun akan lebih efesien bila menggunakan teknik analisis frekuensi.

Keamanan untuk cipherteks hasil modifikasi *Playfair Cipher* dengan *Vigenere Cipher* sangat tinggi. Hal ini disebabkan walaupun panjang kunci dapat diketahui dengan metode Kasiski, cipherteks yang dicoba dipecahkan dengan $26^p$ kemungkinan tidak akan berubah menjadi pesan yang dapat dimengerti. Karena pesan yang sebenarnya sudah dienkripsi dahulu menggunakan *Playfair Cipher* yang menghilangan analisis frekuensi relatif huruf-huruf yang muncul.

## 5 KESIMPULAN

Dari hasil analisis terhadap program hasil modifikasi antara *Playfair Cipher* dengan *Vigenere Cipher* dan keamanan cipherteks, penulis dapat mengambil kesimpulan, yaitu:

1. Program ini mempunyai sebuah kelemahan dalam mengembalikan plainteks ke bentuk awal. Sebelum plainteks dienkripsi, terlebih dahulu akan diatur. Pengaturan ini akan mengakibatkan plainteks kehilangan tanda baca dan spasi, mendapat tambahan huruf X atau Q, dan semua huruf J diganti menjadi huruf I. Setelah plainteks tersebut dienkripsi dan didekripsi kembali, hasil dari pengaturan tersebut tetap ada. Hal ini menyebabkan terkadang plainteks sulit untuk dibaca, terlebih jika plainteks tersebut sangat besar.
2. Kelebihan dari program hasil modifikasi ini terletak pada keamanan cipherteksnya. Keamanan untuk cipherteks hasil modifikasi *Playfair Cipher* dengan *Vigenere Cipher* sangat tinggi. Hal ini disebabkan walaupun panjang kunci dapat diketahui dengan metode Kasiski, cipherteks yang dicoba dipecahkan dengan $26^p$ kemungkinan tidak akan berubah menjadi pesan yang dapat dimengerti. Karena pesan yang sebenarnya sudah dienkripsi dahulu menggunakan *Playfair Cipher* yang menghilangan analisis frekuensi relatif huruf-huruf yang muncul.
3. Terdapat suatu alternatif untuk meningkatkan keamanan dari program hasil modifikasi ini. Yaitu dengan cara menjadikan tabel kunci *Playfair Cipher* sebagai kunci untuk *Vigenere Cipher*. Sehingga panjang kunci *Vigenere Cipher* menjadi 25, dan untuk memecahkan cipherteks dengan metode Kasiski maka jumlah kunci yang harus dicoba adalah $26^{25}$.

## DAFTAR REFERENSI


[1] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, 2006.

[2] Forouzan, Behrouz, *Cryptography and Network Security*, McGraw-Hill, 2008.

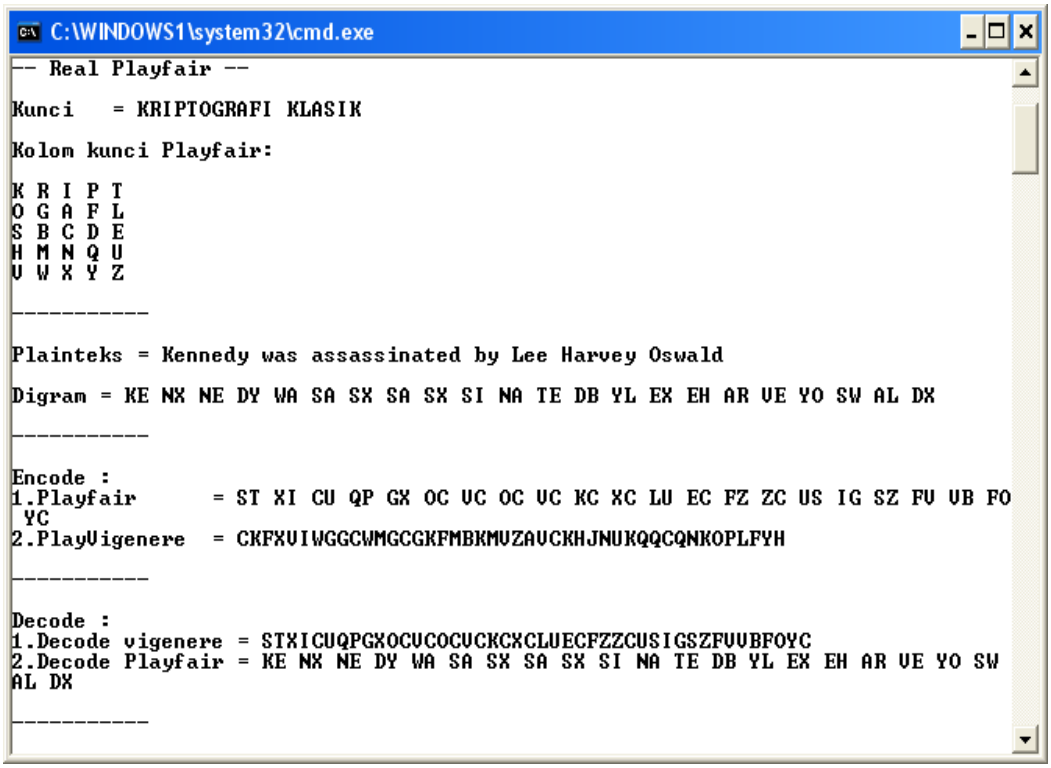
[3] http://en.wikipedia.org/wiki/Playfair_cipher

[4] http://trumpetpower.com/Papers/Crypto/Playfair

[5] http://en.wikipedia.org/wiki/Vigenere_cipher

# LAMPIRAN

## 6 TAMPILAN PROGRAM




**Gambar 1. Hasil program modifikasi**

## 7 SOURCE CODE

### 7.1 Fungsi Pembangkit Tabel Kunci

```
public void setOptionKeyword(String s) {
  //Fungsi pembangkit tabel kunci
        s = s.toUpperCase();
        s = s.replace('J', 'I');
        int realLength = 0;
        for (int i = 0; i < s.length(); i++) {
                // menulis kunci pada barisan atas tabel
                if (s.indexOf(s.charAt(i)) == i) {
                        realLength++;
                        square[i % 5][i / 5] = s.charAt(i);
                }
        }
        int j = 0;
        for (int i = realLength; i < 25; i++) {
                j++;
                if (s.indexOf(alpha.charAt(j)) != -1) {
                        i--;
                } else {
                        square[i % 5][i / 5] = alpha.charAt(j);
                }
        }
        for (int i = 0; i < 5; i++) {
                for (int k = 0; k < 5; k++) {
                        System.out.print(square[k][i]);
                        System.out.print(" ");
                }
                System.out.println("");
        }
  }
```

### 7.2 Fungsi Pengaturan Digram, Enkripsi dan Dekripsi *Playfair Cipher*

```
public String encode(String text) {
 //Fungsi pemanggil enkripsi Playfair
        boolean t = encoder;
        encoder = true;
        String temp = doStuff(text);
        encoder = t;
        return temp;
 }

public String decode(String text) {
//Fungsi pemanggil dekripsi Playfair
        boolean t = encoder;
        encoder = false;
        String temp = doStuff(text);
        encoder = t;
        return temp;
 }

public String doStuff(String s) {
//Fungsi pelaksana pengaturan digram dan proses enkripsi atau dekripsi Playfair
        StringBuffer result = new StringBuffer();
        if (encoder) {
                s = elimWhite(s);
                s = s.toUpperCase();
                if ((s.length() % 2) != 0) {
                        s = s + "X";
                }
                String digram;
                for (int i = 1; i < s.length(); i += 2) {
                        if (s.charAt(i - 1) == s.charAt(i)) {
                                if (s.charAt(i - 1) == 'X') {
                                        digram = s.charAt(i - 1) + "Q";
                                } else {
                                        digram = s.charAt(i - 1) + "X";
                                }
                                System.out.print(digram + " ");
                                i--;
                        } else {
                                digram = ("" + s.charAt(i - 1)) + s.charAt(i);
                                System.out.print(digram + " ");
                        }
                        int[] ind = findDigramIndices(digram);
                        char[] ret = new char[2];
                        if ((ind[0] / 5) == (ind[1] / 5)) {
                                if ((ind[0] % 5) == 4) {
                                        ind[0] = ind[0] - 4;
                                } else {
                                        ind[0]++;
                                }
                                if ((ind[1] % 5) == 4) {
                                        ind[1] = ind[1] - 4;
                                } else {
                                        ind[1]++;
                                }
                                ret[0] = square[ind[0] % 5][ind[0] / 5];
                                ret[1] = square[ind[1] % 5][ind[1] / 5];
                        } else if ((ind[0] % 5) == (ind[1] % 5)) {
                                ind[0] = (ind[0] + 5) % 25;
                                ind[1] = (ind[1] + 5) % 25;
                                ret[0] = square[ind[0] % 5][ind[0] / 5];
                                ret[1] = square[ind[1] % 5][ind[1] / 5];
                        } else {
                                ret[0] = square[ind[0] % 5][ind[1] / 5];
                                ret[1] = square[ind[1] % 5][ind[0] / 5];
                        }
                        result.append(ret[0]);
                        result.append(ret[1]);
                        result.append(" ");
                }
        } else {
                s = elimWhite(s);
                s = s.toUpperCase();
                if ((s.length() % 2) != 0) {
```

```
                        s = s + "X";
                }
                String digram;
                for (int i = 1; i < s.length(); i += 2) {
                        if (s.charAt(i - 1) == s.charAt(i)) {
                                if (s.charAt(i - 1) == 'X') {
                                        digram = s.charAt(i - 1) + "Q";
                                } else {
                                        digram = s.charAt(i - 1) + "X";
                                }
                                i--;
                        } else {
                                digram = ("" + s.charAt(i - 1)) + s.charAt(i);
                        }
                        int[] ind = findDigramIndices(digram);
                        char[] ret = new char[2];
                        if ((ind[0] / 5) == (ind[1] / 5)) {
                                if ((ind[0] % 5) == 0) {
                                        ind[0] = ind[0] + 4;
                                } else {
                                        ind[0]--;
                                }
                                if ((ind[1] % 5) == 0) {
                                        ind[1] = ind[1] + 4;
                                } else {
                                        ind[1]--;
                                }
                                ret[0] = square[ind[0] % 5][ind[0] / 5];
                                ret[1] = square[ind[1] % 5][ind[1] / 5];
                        } else if ((ind[0] % 5) == (ind[1] % 5)) {
                                if ((ind[0] - 5) < 0) {
                                        ind[0] += 25;
                                }
                                if ((ind[1] - 5) < 0) {
                                        ind[1] += 25;
                                }
                                ind[0] = (ind[0] - 5) % 25;
                                ind[1] = (ind[1] - 5) % 25;
                                ret[0] = square[ind[0] % 5][ind[0] / 5];
                                ret[1] = square[ind[1] % 5][ind[1] / 5];
                        } else {
                                ret[0] = square[ind[0] % 5][ind[1] / 5];
                                ret[1] = square[ind[1] % 5][ind[0] / 5];
                        }
                        result.append(ret[0]);
                        result.append(ret[1]);
                        result.append(" ");
                }
        }
        return result.toString();
```

### 7.3 Fungsi Enkripsi dan Dekripsi *Vigenere Cipher*

```
public String encodeVin(String s, String key) {
//Fungsi pemanggil enkripsi Vigenere
        String hasil="";
        s = elimWhite(s);
        key = elimWhite(key);
        key = key.toUpperCase();
        int keylength = key.length();
        // hasil.length() = s.length();
        int Ci;
        int Ki;
        int Pi;
        int j;
        for(int i=0;i<s.length();i++)
        {
                Pi = alpha1.indexOf(s.charAt(i));
                if(i>=keylength) {
                        j = i%keylength;
                } else {
                        j=i;
                }
                Ki = alpha1.indexOf(key.charAt(j));
                Ci =(Pi+Ki)%26;
```

```
                hasil = hasil + alpha1.charAt(Ci);
        }
        return hasil;
 }

public String decodeVin(String s, String key) {
//Fungsi pemanggil dekripsi vigenere
        String hasil="";
        s = elimWhite(s);
        key = elimWhite(key);
        key = key.toUpperCase();
        int keylength = key.length();
        // hasil.length() = s.length();
        int Ci;
        int Ki;
        int Pi;
        int j;
        for(int i=0;i<s.length();i++)
        {
                Ci = alpha1.indexOf(s.charAt(i));
                if(i>=keylength) {
                        j = i%keylength;
                } else {
                        j=i;
                }
                Ki = alpha1.indexOf(key.charAt(j));
                if(Ci<Ki)
                        Ci=Ci+26;
                Pi =(Ci-Ki)%26;
                hasil = hasil + alpha1.charAt(Pi);
        }
        return hasil;
 }
```