***Implementasi Algoritma***

***Vigenere dan Playfair Cipher***

TUGAS KECIL I

IF4020 – Kriptografi



oleh

Wiwit Rifa’i 13513073

Pipin Kurniawati 13513089

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2016

***Source Code* Program**

Program ditulis menggunakan bahasa pemrograman Java. Modul program dibagi berdasarkan jenis algoritma yang direalisasikan menjadi kelas-kelas yang terpisah.

1. **Standard Vigenere Cipher**

/\*\*

\*

\* @class StandardVigenere.java

\*/

public class StandardVigenere **{**

private static final int MAX\_KEY\_LENGTH **=** 25**;**

private String key**,** text**;**

/\*\*

\*

\* constructor

\*/

public StandardVigenere**(**String text**,** String key**)** **{**

**this.**text **=** **new** String**(**text**);**

**this.**key **=** **new** String **(**key**.**toUpperCase**());**

**}**

/\*\*

\*

\* encrypt text based on input key

\*/

public String encrypt**()** **{**

String result **=** **new** String**();**

String plaintext **=** **new** String**(**text**.**toUpperCase**());**

int j **=** 0**;**

**for** **(**int i**=**0**;** i**<**plaintext**.**length**();** i**++)** **{**

char c **=** plaintext**.**charAt**(**i**);**

**if** **((**c **>=** 'A' **&&** c **<=** 'Z'**))** **{** //encrypt the alphabet only

result **+=** **(**char**)** **((**c **+** key**.**charAt**(**j**)** **-** 2 **\*** 'A'**)** **%** 26

**+** 'A'**);**

j **=** **(**j**+**1**)** **%** key**.**length**();**

**}** **else** **if** **(**c**==**' '**)** **{** //to sustain the original format result **+=** c**;**

**}**

**}**

**return** result**;**

**}**

/\*\*

\*

\* decrypt text based on input key

\*/

public String decrypt**()** **{**

String result **=** **new** String**();**

String ciphertext **=** **new** String**(**text**.**toUpperCase**());**

int j **=** 0**;**

**for** **(**int i**=**0**;** i**<**ciphertext**.**length**();** i**++)** **{**

char c **=** ciphertext**.**charAt**(**i**);**

**if** **(**c **>=** 'A' **&&** c **<=** 'Z'**)** **{** //encrypt the alphabet only

result **+=** **(**char**)** **((**c **-** key**.**charAt**(**j**)** **+** 26**)** **%** 26**+**'A'**);**

j **=** **(**j**+**1**)** **%** key**.**length**();**

**}** **else** **if** **(**c**==**' '**)** **{**

result **+=** c**;**

**}**

**}**

**return** result**.**toLowerCase**();**

**}**

**}**

1. **Extended Vigenere Cipher**

/\*\*

\*

\* @file ExtendedVigenere.java

\*/

public class ExtendedVigenere **{**

private static final int MAX\_KEY\_LENGTH **=** 25**;**

private String key**,** text**;**

/\*\*

\*

\* @param text

\* @param key

\* constructor

\*/

public ExtendedVigenere**(**String text**,** String key**)** **{**

**this.**text **=** **new** String**(**text**);**

**this.**key **=** **new** String **(**key**);**

**}**

/\*\*

\*

\* @return string

\* encrypt text based on input key

\*/

public String encrypt**()** **{**

String result **=** **new** String**();**

String plaintext **=** **new** String**(**text**);**

**for** **(**int i**=**0**;** i**<**plaintext**.**length**();** i**++)** **{**

char c **=** plaintext**.**charAt**(**i**);**

result **+=** **(**char**)** **((**c **+** key**.**charAt**(**i**%**key**.**length**()))%**256**);**

**}**

**return** result**;**

**}**

/\*\*

\*

\* @return string

\* dectypt text based on input key

\*/

public String decrypt**()** **{**

String result **=** **new** String**();**

String ciphertext **=** **new** String**(**text**);**

**for** **(**int i**=**0**;** i**<**ciphertext**.**length**();** i**++)** **{**

char c **=** ciphertext**.**charAt**(**i**);**

result **+=** **(**char**)** **((**c **-** key**.**charAt**(**i**%**key**.**length**())** **+** 256**)**

**%** 256**);**

**}**

**return** result**;**

**}**

**}**

1. **Modified Vigenere Cipher**
2. **Playfair Cipher**

/\*\*

\*

\* @file PlayfairCipher.java

\*/

**import** java**.**util**.**Vector**;**

public class PlayfairCipher **{**

private static final int MAX\_KEY\_LENGTH **=** 25**;**

private char**[][]** key**;**

private String text**;**

/\*\*

\*

\* @param text

\* @param keyInput

\* ctor

\*/

public PlayfairCipher**(**String text**,** Vector**<**String**>** keyInput**)** **{**

**this.**text **=** **new** String**(**text**.**toUpperCase**());**

**this.**key **=** **new** char**[**5**][**5**];**

**for** **(**int i**=**0**;** i**<**5**;** i**++)** **{**

String line **=** keyInput**.**get**(**i**).**replaceAll**(**"\\s+"**,**""**);**

char**[]** splited **=** line**.**toUpperCase**().**toCharArray**();**

**for** **(**int j**=**0**;** j**<**5**;** j**++)** **{**

key**[**i**][**j**]** **=** splited**[**j**];**

**}**

**}**

**}**

/\*\*

\*

\* @param text

\* @return string

\* return string alphabet only

\*/

public String clearSymbols**(**String text**)** **{**

String result **=** **new** String**();**

**for** **(**int i**=**0**;** i**<**text**.**length**();** i**++)** **{**

**if** **(**text**.**charAt**(**i**)** **<** 'A' **||** text**.**charAt**(**i**)** **>** 'Z'**)** **{**

**continue;**

**}** **else** **{**

result **+=** **(**char**)** **(**text**.**charAt**(**i**));**

**}**

**}**

**return** result**;**

**}**

/\*\*

\*

\* @return string

\* encryption function

\*/

public String encrypt**()** **{**

String ciphertext **=** **new** String**();**

text **=** clearSymbols**(**text**);**

text **=** text**.**replaceAll**(**"J"**,**"I"**);**

//replace second character with Z

**for** **(**int i**=**0**;** i**<**text**.**length**()-**1**;** i**+=**2**)** **{**

**if** **(**text**.**charAt**(**i**)==**text**.**charAt**(**i**+**1**))** **{**

text **=** **new** StringBuffer**(**text**).**insert**(**i**+**1**,**

"Z"**).**toString**();**

**}**

**}**

**if** **(**text**.**length**()** **%** 2 **!=** 0**)** **{**

text **=** **new** StringBuffer**(**text**).**insert**(**text**.**length**(),**

"Z"**).**toString**();**

**}**

**for** **(**int i**=**0**;** i**<**text**.**length**()-**1**;** i**+=**2**)** **{**

int col1**=**0**,** row1**=**0**,** col2**=**0**,** row2**=**0**;**

**for** **(**int j**=**0**;** j**<**key**.**length**;** j**++)** **{**

**for** **(**int k**=**0**;** k**<**key**[**j**].**length**;** k**++)** **{**

**if** **(**key**[**j**][**k**]** **==** text**.**charAt**(**i**))** **{**

row1**=**j**;** col1 **=** k**;**

**}**

**if** **(**key**[**j**][**k**]** **==** text**.**charAt**(**i**+**1**))** **{**

row2**=**j**;** col2 **=** k**;**

**}**

**}**

**}**

//substitution

**if** **(**row1**==**row2**)** **{**

ciphertext **+=** key**[**row1**][(**col1**+**1**)%**5**];**

ciphertext **+=** key**[**row1**][(**col2**+**1**)%**5**];**

**}** **else** **if** **(**col1**==**col2**)** **{**

ciphertext **+=** key**[(**row1**+**1**)%**5**][**col1**];**

ciphertext **+=** key**[(**row2**+**1**)%**5**][**col1**];**

**}** **else** **{**

ciphertext **+=** key**[**row1**][**col2**];**

ciphertext **+=** key**[**row2**][**col1**];**

**}**

**}**

**return** ciphertext**;**

**}**

//decryption function

public String decrypt**()** **{**

String plaintext **=** **new** String**();**

text **=** clearSymbols**(**text**);**

**for** **(**int i**=**0**;** i**<**text**.**length**()-**1**;** i**+=**2**)** **{**

int col1**=**0**,** row1**=**0**,** col2**=**0**,** row2**=**0**;**

**for** **(**int j**=**0**;** j**<**key**.**length**;** j**++)** **{**

**for** **(**int k**=**0**;** k**<**key**[**j**].**length**;** k**++)** **{**

**if** **(**key**[**j**][**k**]** **==** text**.**charAt**(**i**))** **{**

row1**=**j**;** col1 **=** k**;**

**}**

**if** **(**key**[**j**][**k**]** **==** text**.**charAt**(**i**+**1**))** **{**

row2**=**j**;** col2 **=** k**;**

**}**

**}**

**}**

**if** **(**row1**==**row2**)** **{**

**if** **(**col1**==**0**)** **{**

col1**=**5**;**

**}** **else** **if** **(**col2**==**0**)** **{**

col2**=**5**;**

**}**

plaintext **+=** key**[**row1**][**col1**-**1**];**

plaintext **+=** key**[**row1**][**col2**-**1**];**

**}** **else** **if** **(**col1**==**col2**)** **{**

**if** **(**row1**==**0**)** **{**

row1**=**4**;**

**}** **else** **if** **(**row2**==**0**)** **{**

row2**=**4**;**

**}**

plaintext **+=** key**[**row1**-**1**][**col1**];**

plaintext **+=** key**[**row2**-**1**][**col1**];**

**}** **else** **{**

plaintext **+=** key**[**row1**][**col2**];**

plaintext **+=** key**[**row2**][**col1**];**

**}**

**}**

**return** plaintext**;**

**}**

**}**

**Tampilan Antarmuka Program**

**Contoh Plainteks dan Cipherteks**

1. **Contoh Plainteks**

Plainteks yang digunakan sebagai masukan program memiliki 3 ukuran kategori, yaitu yang berukuran kecil, sedang, dan besar. Program dirancang untuk mampu membaca plainteks dari console maupun dari file (.txt, .docx, .pdf, dll) dengan cara membaca byte dari file tersebut. Berikut contoh plainteks untuk masing-masing kategori: