

rail-fence cipher

12th project



By:

Perihan Hassan (17105545)

Engy Elghetany (17105762)

Omar Dawaba (17105423)

Yahia Moustafa (17104718)

Seif Bakry (17106233)

Submitted to:

ENG. Nagham Ayman

**Introduction:**

Rail-fence is one of the transposition techniques that mixes up the request for the letters of a message utilizing an essential calculation. The rail fence figure works by composing your message on exchange lines over the page, and afterward perusing off each line thus.

What the system will do?

We made a GUI that will take the given plain-text message and a numeric key, encrypt/decrypt the given text using Rail Fence algorithm.

**Algorithm:**

Encryption:

1. The plain text is written down on a sequence of columns according to the depth (no. of rows)
2. Cipher text is read off as a sequence of rows.

Decryption:

1. The plain text is written down on a sequence of rows
2. Then read off as a sequence of columns

The Rail fence Cipher is easy to use to jumble up a message very quickly. It has some problems depending on what route you pick as you can end up with large chunks of plaintext in the right order within the cipher text (or simply reversed) which gives away a lot about the width of the grid and the route used.

An historical use of the rail fence Cipher was the Union rail fence Cipher used by the Union forces during the American Civil War. Rather than transposing letters by the given route, it moved whole words around.

**Implementation:**

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

/\*\*

\*

\* @author Dawaba

\*/

public class Secuirty\_project extends Application {

@Override

public void start(Stage primaryStage) {

Button btn\_enc = new Button();

Button btn\_dec = new Button();

Label key\_text = new Label("Enter Key");

TextField key\_field = new TextField();

TextField enc\_field = new TextField();

enc\_field.setMaxWidth(250);

TextField dec\_field = new TextField();

dec\_field.setMaxWidth(250);

btn\_enc.setText("encrypt");

btn\_enc.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

String key = key\_field.getText();

int numRails = Integer.parseInt(key);

String data = enc\_field.getText();

char[] encrypted = new char[data.length()];

int n = 0;

rail\_fence rf = new rail\_fence();

rf.input = data;

rf.key=numRails;

String enrypted\_word = rf.encryption();

dec\_field.setText(enrypted\_word);

}

});

btn\_dec.setText("decrypt");

btn\_dec.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

String key = key\_field.getText();

int numRails = Integer.parseInt(key);

String data = dec\_field.getText();

char[] encrypted = new char[data.length()];

int n = 0;

rail\_fence rf = new rail\_fence();

rf.input = data;

rf.key=numRails;

String enrypted\_word = rf.decryption();

enc\_field.setText(enrypted\_word);

}

});

VBox hroot = new VBox();

hroot.setAlignment(Pos.CENTER);

hroot.setSpacing(20);

GridPane root = new GridPane();

root.setAlignment(Pos.CENTER);

root.setHgap(20);

root.add(key\_text, 0, 0);

root.add(key\_field, 1, 0);

hroot.getChildren().add(root);

hroot.getChildren().add(enc\_field);

hroot.getChildren().add(btn\_enc);

hroot.getChildren().add(dec\_field);

hroot.getChildren().add(btn\_dec);

Scene scene = new Scene(hroot, 300, 250);

primaryStage.setTitle("Secuirty Project");

primaryStage.setScene(scene);

primaryStage.show();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

}

public class rail\_fence {

int key;

String input;

public String encryption()

{

char[] input\_array = input.toCharArray();

char[] output = new char[input.length()];

int i = 0 , j = 0;

int length = input.length();

int x = input.length()% 2;

if(x==1)

{

while(j < input.length()){

output[j] = input\_array[i];

j++;

i = (i+key)%input.length();

}

}

else {

while(j < input.length()){

output[j] = input\_array[i];

j++;

if(i == key){

i = (i+key)%(input.length());

i++;

}

else if(i < key)

{

i = (i+key)%(input.length());

}

else

{

i=1+i;

i = (i+key)%(input.length());

}

}

}

return new String(output) ;

}

public String decryption()

{

int length = input.length();

int ylength=(int) Math.ceil((double)length/key);

char[] input\_array = input.toCharArray();

char[] output = new char[input.length()];

int i = 0 , j = 0;

int x = input.length()% 2;

if(x==1){

while(j < input.length()){

output[j] = input\_array[i];

j++;

i = (i+ylength)%input.length();

}

}

else {

while(j < input.length()){

output[j] = input\_array[i];

j++;

if(i == key){

i = (i+ylength)%(input.length());

}

else if(i < key)

{

i = (i+ylength)%(input.length());

}

else

{

i=i+1;

i = (i+ylength)%(input.length());

}

}

}

return new String(output);

}

}

**Screenshots:**





