|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 1** | **Play Fair Cipher** | | |
| Date of Exercise | 16 – 12 - 2014 | Date of Output Verification | 13 – 01 – 2015 |

**Question**

1. Implement a server client communication with play fair cipher for encryption and decryption of data sent.

**Procedure**

* Remove any punctuation or characters that are not present in the key square (this may mean spelling out numbers, punctuation etc.).
* Identify any double letters in the plaintext and replace the second occurence with an 'x' e.g. 'hammer' -> 'hamxer'.
* If the plaintext has an odd number of characters, append an 'x' to the end to make it even.
* Break the plaintext into pairs of letters, e.g. 'hamxer' -> 'ha mx er'
* The algorithm now works on each of the letter pairs.
* Locate the letters in the key square, (the examples given are using the key square above)
  + If the letters are in different rows and columns, replace the pair with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter. 'ha' -> 'bo', 'es' -> 'il'
  + If the letters appear on the same row of the table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row). 'ma' -> 'or', 'lp' -> 'pq'
  + If the letters appear on the same column of the table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column). 'rk' -> 'dt', 'pv' -> 'vo'

**Program**

Server:

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class NetworkSecurityLabExp1dPlayFairCipherServer {

public static Scanner in = new Scanner(System.in);

public static String key, keycpy, input, inputcpy, output = "", alph, msg = "";

public static char m[][] = new char[5][5], l = 'a';

public static boolean keyid = false;

public static int pa, pb, qa, qb;

public static void main(String[] args) throws IOException {

System.out.println("\n------Server------\nWaiting for Client to connect!");

ServerSocket ss = new ServerSocket(3000);

Socket s = ss.accept();

System.out.println("\nClient Connected!");

OutputStream os = s.getOutputStream();

PrintWriter pw = new PrintWriter(os, true);

setkey();

encode();

pw.println(output);

System.out.println("\nSent the Encoded text to Client");

}

public static void setkey() {

int p = 0;

System.out.println("\n--------Set Key--------");

System.out.print("\nEnter Key: ");

do {

p = 0;

key = in.nextLine();

key = key.replaceAll(" ", "");

for (char a : key.toCharArray()) {

if (((int) a >= 65 && (int) a <= 90) || ((int) a >= 97 && (int) a <= 122)) {

} else {

p = 1;

}

}

if (p == 1) {

System.out.println("\nNote: Only Alphabets are Allowed");

System.out.print("\nEnter Valid key: ");

}

} while (p == 1);

key = key.toLowerCase();

key = key.replaceAll("j", "i");

keycpy = key;

keyid = false;

alph = "abcdefghiklmnopqrstuvwxyz";

setmatrix();

}

public static void encode() {

System.out.println("\n---------Encode--------");

texttoencode();

for (int i = 0; i < inputcpy.length(); i += 2) {

match(inputcpy.charAt(i), inputcpy.charAt(i + 1));

if (pa == qa) {

if (pb == 4 || qb == 4) {

if (pb == 4) {

output = output + m[pa][0];

} else {

output = output + m[pa][pb + 1];

}

if (qb == 4) {

output = output + m[qa][0];

} else {

output = output + m[qa][qb + 1];

}

} else {

output = output + m[pa][pb + 1] + m[qa][qb + 1];

}

} else if (pb == qb) {

if (pa == 4 || qa == 4) {

if (pa == 4) {

output = output + m[0][pb];

} else {

output = output + m[pa + 1][pb];

}

if (qa == 4) {

output = output + m[0][qb];

} else {

output = output + m[qa + 1][qb];

}

} else {

output = output + m[pa + 1][pb] + m[qa + 1][qb];

}

} else {

output = output + m[pa][qb] + m[qa][pb];

}

}

System.out.println("Encoded Text: " + output);

}

public static void match(char a, char b) {

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

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

if (m[i][j] == a) {

pa = i;

pb = j;

}

if (m[i][j] == b) {

qa = i;

qb = j;

}

}

}

}

public static void texttoencode() {

output = "";

System.out.print("\nEnter Text: ");

input = in.nextLine();

inputcpy = input.replaceAll("j", "i");

for (int i = 0; i < inputcpy.length() - 1; i++) {

if (inputcpy.charAt(i) == inputcpy.charAt(i + 1)) {

inputcpy = inputcpy.substring(0, i + 1) + "x" + inputcpy.substring(i + 1, inputcpy.length());

}

}

if (inputcpy.length() % 2 == 1) {

inputcpy += "x";

}

System.out.println("Manipulated Text: " + inputcpy);

}

public static void setmatrix() {

System.out.println("\nKey Matrix");

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

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

m[i][j] = fill();

System.out.print(" " + m[i][j]);

}

System.out.println("");

}

}

public static char fill() {

if (keyid == false) {

return keychar();

} else {

return alphchar();

}

}

public static char keychar() {

char t = 0;

int p = 0;

for (char c : keycpy.toCharArray()) {

if (c != '1') {

t = c;

keycpy = keycpy.replaceAll(c + "", "1");

alph = alph.replaceAll(c + "", "1");

p = 1;

return t;

}

}

if (p == 0) {

keyid = true;

return alphchar();

}

return 0;

}

public static char alphchar() {

char t = 0;

for (char c : alph.toCharArray()) {

if (c != '1') {

t = c;

alph = alph.replaceAll(c + "", "1");

return t;

}

}

return 0;

}

}

Client:

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class NetworkSecurityLabExp1dPlayFairCipherClient {

public static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

public static Scanner in = new Scanner(System.in);

public static String key, keycpy, input, inputcpy, output = "", alph;

public static char m[][] = new char[5][5], l = 'a';

public static boolean keyid = false;

public static int pa, pb, qa, qb;

public static void main(String[] args) throws IOException {

System.out.println("\n------Client------");

Socket s = new Socket("127.0.0.1", 3000);

System.out.println("\nConnected to Server\n");

setkey();

InputStream is = s.getInputStream();

BufferedReader brs = new BufferedReader(new InputStreamReader(is));

System.out.println("\nWaiting for Server to Send CipherText");

input = brs.readLine();

System.out.println("Received CipherText: " + input);

decode();

System.out.println();

}

public static void setkey() {

int p = 0;

System.out.println("\n--------Set Key--------");

System.out.print("\nEnter Key: ");

do {

p = 0;

key = in.nextLine();

key = key.replaceAll(" ", "");

for (char a : key.toCharArray()) {

if (((int) a >= 65 && (int) a <= 90) || ((int) a >= 97 && (int) a <= 122)) {

} else {

p = 1;

}

}

if (p == 1) {

System.out.println("\nNote: Only Alphabets are Allowed");

System.out.print("\nEnter Valid key: ");

}

} while (p == 1);

key = key.toLowerCase();

key = key.replaceAll("j", "i");

keycpy = key;

keyid = false;

alph = "abcdefghiklmnopqrstuvwxyz";

setmatrix();

}

public static void match(char a, char b) {

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

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

if (m[i][j] == a) {

pa = i;

pb = j;

}

if (m[i][j] == b) {

qa = i;

qb = j;

}

}

}

}

public static void decode() {

System.out.println("\n---------Decode--------");

inputcpy = input;

for (int i = 0; i < inputcpy.length(); i += 2) {

match(inputcpy.charAt(i), inputcpy.charAt(i + 1));

if (pa == qa) {

if (pb == 0 || qb == 0) {

if (pb == 0) {

output = output + m[pa][4];

} else {

output = output + m[pa][pb - 1];

}

if (qb == 0) {

output = output + m[qa][4];

} else {

output = output + m[qa][qb - 1];

}

} else {

output = output + m[pa][pb - 1] + m[qa][qb - 1];

}

} else if (pb == qb) {

if (pa == 0 || qa == 0) {

if (pa == 0) {

output = output + m[4][pb];

} else {

output = output + m[pa - 1][pb];

}

if (qa == 0) {

output = output + m[4][qb];

} else {

output = output + m[qa - 1][qb];

}

} else {

output = output + m[pa - 1][pb] + m[qa - 1][qb];

}

} else {

output = output + m[pa][qb] + m[qa][pb];

}

}

System.out.print("Decoded Text: " + output);

}

public static void setmatrix() {

System.out.println("\nKey Matrix");

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

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

m[i][j] = fill();

System.out.print(" " + m[i][j]);

}

System.out.println("");

}

}

public static char fill() {

if (keyid == false) {

return keychar();

} else {

return alphchar();

}

}

public static char keychar() {

char t = 0;

int p = 0;

for (char c : keycpy.toCharArray()) {

if (c != '1') {

t = c;

keycpy = keycpy.replaceAll(c + "", "1");

alph = alph.replaceAll(c + "", "1");

p = 1;

return t;

}

}

if (p == 0) {

keyid = true;

return alphchar();

}

return 0;

}

public static char alphchar() {

char t = 0;

for (char c : alph.toCharArray()) {

if (c != '1') {

t = c;

alph = alph.replaceAll(c + "", "1");

return t;

}

}

return 0;

}

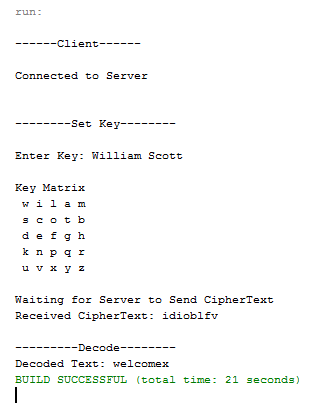
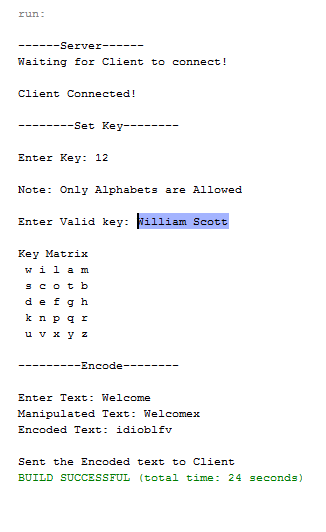
}

**Input**

Key: William Scott

Plain Text: welcome

**Output**



**Result**

The given Play Fair cipher is successfully implemented in a client server communication.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Manoj Kumar

Date: 13-01-2015