**INFORMATION AND NETWORK SECURITY COURSE**

**Classic Encryption Project**



**Member of Group:**

Dely Teja Mukti 05111540000003

Yoga Samudra 05111540000080

M. Ilham Febriyanto 05111540000083

**Lecturer:**

Henning Titi Ciptaningtyas, S.Kom, M.Kom

**DEPARTMENT OF INFORMATICS**

**FACULTY OF INFORMATION TECHNOLOGY**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**2018**

## 1. CAESAR CIPHER

## DEFINITION

Caesar Cipher is the simplest method of message encryption found by Julius Caesar. It make use of substitution cipher which involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example, shifting the alphabet 5 letter next to it, so letter A become F, B become G, C become H, and etc.

As unreadable as the resulting ciphertext may appear, the Caesar Cipher is one of the weakest forms of encryption one can employ for the following reasons:

* The key space is very small. Using a [brute force attack](https://learncryptography.com/attack-vectors/brute-force-attack) method, one could easily try all (25) possible combinations to decrypt the message without initially knowing the key.
* The structure of the original plaintext remains intact. This makes the encryption method vulnerable to [frequency analysis](https://learncryptography.com/attack-vectors/frequency-analysis) - by looking at h ow often certain characters or sequences of characters appear, one can discover patterns and potentially discover the key without having to perform a full brute force search.

## DESCRIPTION

In this project, we use Javascript to make the encryption and decryption code and combine it with web programming. The input type is text form (read as string) in html and the output type is also in text form.

## SOURCE CODE

File Name : caesar.php

Function : caesarShift

Input : text (string), shift (integer)

Output : result (string)

Source code

var result = "";

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

var c = text.charCodeAt(i);

if (65 <= c && c <= 90) result += String.fromCharCode((c - 65 + shift) % 26 + 65); // Uppercase

else if (97 <= c && c <= 122) result += String.fromCharCode((c - 97 + shift) % 26 + 97); // Lowercase

else result += text.charAt(i); // Copy

}

return result;

File Name : caesar.php

Function : doCrypt

Input : isDecrypt (boolean)

Output : textElem (string)

Source code

var shiftText = AssignKey();

if (!/^-?\d+$/.test(shiftText)) {

alert("Shift is not an integer");

return;

}

var shift = parseInt(shiftText, 10);

if (shift < 0 || shift >= 26) {

alert("Shift is out of range");

return;

}

if (isDecrypt)

shift = (26 - shift) % 26;

var textElem = AssignPlain();

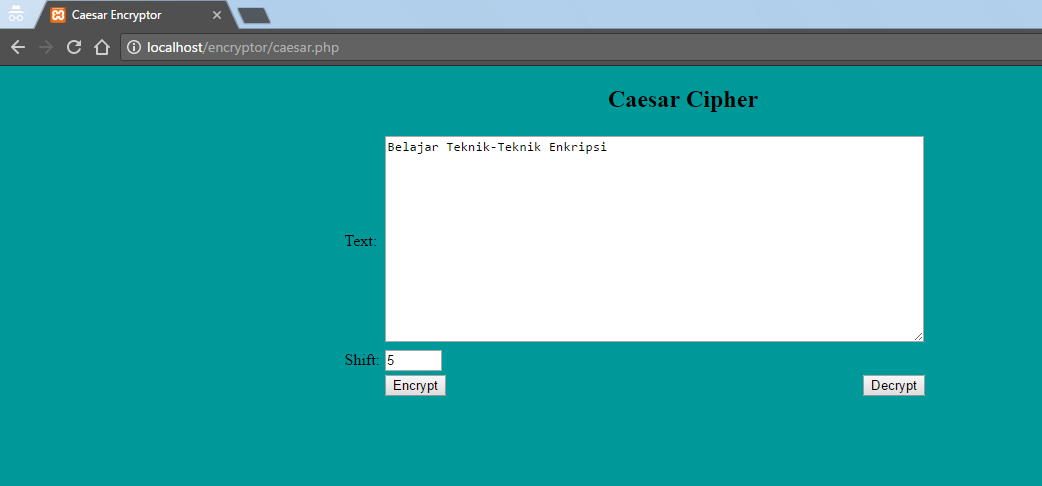
textElem = caesarShift(textElem, shift);

console.log(textElem);

document.getElementById("plain").value = textElem;

## IMPLEMENTATION

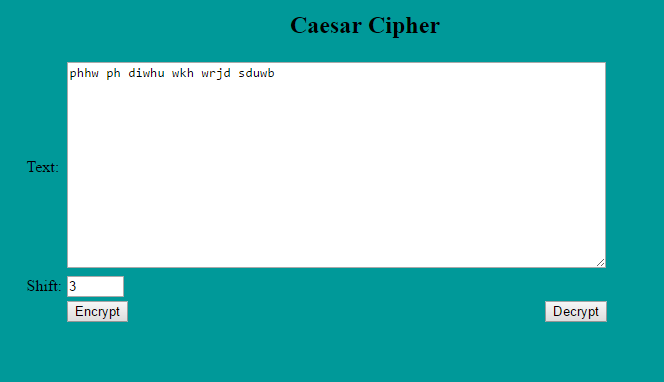
1. Open the php file (localhost/encryption/caesar.php)



2. Write the plain text that we want to encrypt in the given space labeled “Text”, for example : meet me after the toga party. Also input how much shift we want to use in the given space labeled “Shift”, for example we want the encryption to be shifting 3 alphabet, so we input 3



3. After giving both input, click Encrypt button to encrypt the plain text. And will resulting in the following picture.



4. To decrypt it back, give input to the space labeled “Text” with the encrypted text. And the shift key must be the same as the one being used when encrypting the message. From the example above, we use the shifting 3 keys forward, so the input in shift key should also be 3. After that, click Decrypt.



5. It would back to the plain text it should be.

## EXPERIMENTAL RESULT

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Input File / Data** | **Key** | **Output File / Data** |
|
| 1. | meet me after the toga party | C = E(3, p) = (p + 3) mod 26 | PHHW PH DIWHU WKH WRJD SDUWB |
| 2. | Information And Network Security Course | C = E(5, p) = (p + 5) mod 26 | Nsktwrfynts Fsi Sjybtwp Xjhzwnyd Htzwxj |
| 3. | Dely Yoga dan Yanto | C = E(25, p) = (p + 25) mod 26 | Cdkx Xnfz czm Xzmsn |
| 4. | Informatika C1F ITS | C = E(6, p) = (p + 6) mod 26 | Otluxsgzoqg I1L OZY |

## REFERENCE(S)

<http://practicalcryptography.com/ciphers/simple-substitution-cipher/> (source code)

<https://www.nayuki.io/page/caesar-cipher-javascript> (source code)

<https://learncryptography.com/classical-encryption/caesar-cipher> (definition)

## 2. MONOALPHABETIC CIPHER

## DEFINITION

Monoalphabetic Cipher or usually called as Substitution Cipher is an encryption method which can be demonstrated by writing out the alphabet in some order to represent the substitution. The cipher alphabet may be shifted or reversed (creating the [Caesar](https://en.wikipedia.org/wiki/Caesar_cipher) and [Atbash](https://en.wikipedia.org/wiki/Atbash) ciphers, respectively) or scrambled in a more complex fashion, in which case it is called a **mixed alphabet** or **deranged alphabet**. Traditionally, mixed alphabets may be created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet in the usual order.

## DESCRIPTION

In this project, we use Javascript to make the encryption and decryption code and combine it with web programming. The input type is text form (read as string) in html and the output type is also in text form.

## SOURCE CODE

File Name : mono.php

Function : monoCrypt

Input :

Output : ciphertext (string)

Source code

var text=AssignPlain();

var key = AssignKey();

var plaintext = text.toLowerCase();

var ekey = key.toLowerCase().replace(/[^a-z]/g,"");

if(plaintext.length < 1){ alert("please enter some plaintext (letters and numbers only)"); return; }

if(ekey.length != 26){ alert("key must be 26 characters in length"); return; }

var ciphertext = ""; var re = /[a-z]/;

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

if(re.test(plaintext.charAt(i))) ciphertext += ekey.charAt(plaintext.charCodeAt(i)-97);

else ciphertext += plaintext.charAt(i);

}

document.getElementById("plain").value = ciphertext.toUpperCase();

File Name : mono.php

Function : demonoCrypt

Input :

Output : plaintext (string)

Source code

var text=AssignPlain();

var key = AssignKey();

var ciphertext = text.toLowerCase();

var ekey = key.toLowerCase().replace(/[^a-z]/g, "");

if(ciphertext.length < 1){ alert("please enter some ciphertext (letters only)"); return; }

if(ekey.length != 26){ alert("key must be 26 characters in length"); return; }

var plaintext = ""; var re = /[a-z]/;

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

if(re.test(ciphertext.charAt(i))) plaintext += String.fromCharCode(ekey.indexOf(ciphertext.charAt(i))+97);

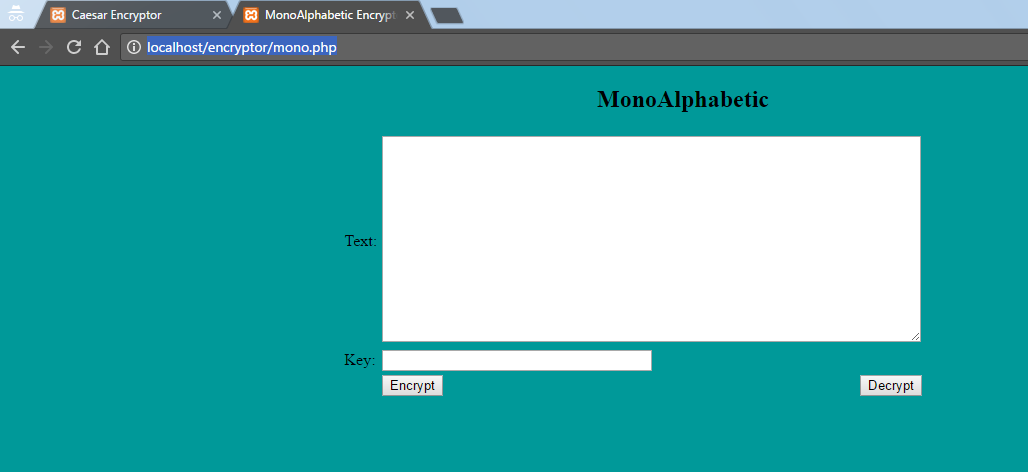
else plaintext += ciphertext.charAt(i);

}

document.getElementById("plain").value = plaintext.toUpperCase();

## IMPLEMENTATION

1. Open the php file (localhost/encryption/mono.php)



2. Write the plain text that we want to encrypt in the given space labeled “Text”, for example like the one in the pdf book. Also input the 26 digit alphabet encryption key to replace normal alphabet in order (abcdefghijklmnopqrstuvwxyz) we want to use in the given space labeled “Shift”, for example we want the encryption key to be “bmiyngocdvhaeukxzjtwlsqfrp”



3. After giving both input, click Encrypt button to encrypt the plain text. And will resulting in the following picture.



4. To decrypt it back, give input to the space labeled “Text” with the encrypted text. And the decryption key must be the same as the one being used when encrypting the message. After that, click Decrypt.



5. It would back to the plain text it should be.

## EXPERIMENTAL RESULT

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Input File / Data** | **Key** | **Output File / Data** |
|
| 1. | it was disclosed yesterday that several informal but  direct contacts have been made with political  representatives of the viet cong in moscow | C = bmiyngocdvhaeukxzjtwlsqfrp | DW QBT YDTIAKTNY RNTWNJYBR WCBW TNSNJBA DUGKJEBA MLW  YDJNIW IKUWBIWT CBSN MNNU EBYN QDWC XKADWDIBA  JNXJNTNUWBWDSNT KG WCN SDNW IKUO DU EKTIKQ |
| 2. | Dely Yoga dan Yanto | C = zyxwvutsrqponmlkjihgfedcba | WVOB BLTZ WZM BZMGL |
| 3. | Keamanan Informasi dan Jaringan | C = zyxwvutsrqponmlkjihgfedcba | PVZNZMZM RMULINZHR WZM QZIRMTZM |
| 4. | Informatika C1F ITS | C = qwertyuiopasdfghjklzxcvbnm | OFYGKDQZOAQ E1Y OZL |

## REFERENCE(S)

<http://practicalcryptography.com/ciphers/simple-substitution-cipher/> (source code)

<https://en.wikipedia.org/wiki/Substitution_cipher> (definition)

## 3. PLAYFAIR CIPHER

## DEFINITION

PlayFair Cipher is the best-known multiple-letter encryption cipher which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The PlayFair algorithm is based on the use of a 5 \* 5 matrix of letters constructed using a keyword that will act as the key for encrypting your plaintext. Each of the 25 letters must be unique and one letter of the alphabet (usually Q) is omitted from the table (as there are 25 spots and 26 letters in the alphabet).

## DESCRIPTION

In this project, we use javascript to make the encryption and decryption code and combine it with web programming. The input type is text form (read as string) in html and the output type is also in text form.

## SOURCE CODE

### PlayfairCipher.js

File Name : playfair.php

Function : setKey

Input : key (string)

Output : grid (two dimensional array)

Source code

if (key) {

// create grid from key

var alphabet = ['abcdefghiklmnopqrstuvwxyz'];

var sanitizedKey = key.toLowerCase().replace(/j/g, 'i').replace(/[^a-z]/g, '');

var keyGrid = [...new Set(`${sanitizedKey}${alphabet}`)];

grid = [];

for (let i = 0; i < keyGrid.length; i += 5) {

grid.push(keyGrid.slice(i, i + 5));

}

}

return grid;

File Name : playfair.php

Function : preProcess

Input : input (string), decrypt (boolean)

Output : ciphertext (string)

Source code

// split into duples, fixing double-letters (hello => he lx lo) and padding

text = input.toLowerCase().replace(/[^a-z]/g, '').replace(/j/g, 'i').split('').filter(x => x !== ' ');

duples = [];

for (let i = 0; i < text.length; i += 2) {

currentDuple = text.slice(i, i + 2);

if (!decrypt && currentDuple.length !== 2) {

currentDuple.push('x');

duples.push(currentDuple);

}else if (!decrypt && currentDuple[0] === currentDuple[1]) {

text.splice(i + 1, 0, 'x');

duples.push(text.slice(i, i + 2));

}else {

duples.push(currentDuple);

}

// console.log(currentDuple)

}

// find row and column for each letter in duple

coordinates = [];

duples.forEach((duple) => {

coordinates.push(duple.map((letter) => {

let col;

row = grid.findIndex(row => {

rowIdx = row.findIndex(x => x === letter);

if (rowIdx >= 0) {

col = rowIdx;

return true;

}

return false;

});

console.log(row, col)

return [row, col];

}));

});

return coordinates;

File Name : playfair.php

Function : doPlayfair

Input : decrypt (boolean)

Output : processedText (string)

Source code

var input = AssignPlain();

var grid = setKey(AssignKey());

// console.log(grid);

if (!grid) return 'First set the key!';

if (input && decrypt && input.length % 2 !== 0) input += 'x';

const coordinates = preProcess(input, decrypt);

// set modifiers to respond appropriately based on decrypt switch

// set modifiers to respond appropriately based on decrypt switch

const modifier = decrypt ? -1 : 1;

const wall = decrypt ? 0 : 4;

const phase = decrypt ? 4 : -4;

const processedLocs = [];

coordinates.forEach((loc) => {

// loc: [ [ firstLetterR, firstLetterC ], [ secondLetterR, secondLetterC ] ]

// modified: [ [ newFirstLetterR, newFirstLetterC ], [ newSecondLetterR, newSecondLetter R ] ]

let modifiedLoc = [];

// handle letters on the same row

if (loc[0][0] === loc[1][0]) {

// increment/decrement the column

modifiedLoc[0] = loc[0][1] === wall ? [loc[0][0], wall + phase] : [loc[0][0], loc[0][1] + modifier];

modifiedLoc[1] = loc[1][1] === wall ? [loc[1][0], wall + phase] : [loc[1][0], loc[1][1] + modifier];

return processedLocs.push(modifiedLoc);

}

// handle letters in the same column

if (loc[0][1] === loc[1][1]) {

// increment/decrement the row

modifiedLoc[0] = loc[0][0] === wall ? [wall + phase, loc[0][1]] : [loc[0][0] + modifier, loc[0][1]];

modifiedLoc[1] = loc[1][0] === wall ? [wall + phase, loc[1][1]] : [loc[1][0] + modifier, loc[1][1]];

return processedLocs.push(modifiedLoc);

}

// handle different rows, different columns

modifiedLoc[0] = [loc[0][0], loc[1][1]];

modifiedLoc[1] = [loc[1][0], loc[0][1]];

processedLocs.push(modifiedLoc);

});

// translate coordinates into ciphertext

const processedText = processedLocs

.map((loc) => [grid[loc[0][0]][loc[0][1]], grid[loc[1][0]][loc[1][1]]].join(''))

.join('');

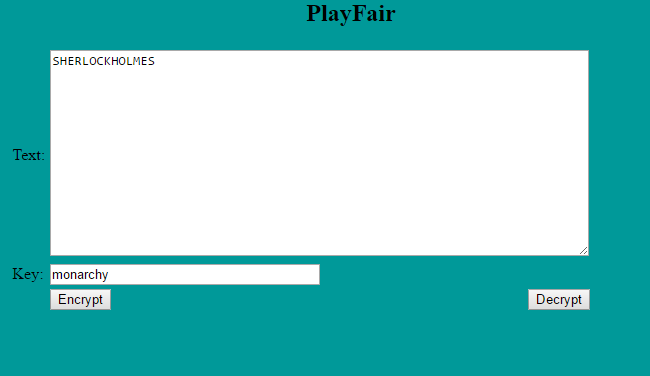
document.getElementById("plain").value = processedText.toUpperCase();

## IMPLEMENTATION

1. Open the php file (localhost/encryption/playfair.php)



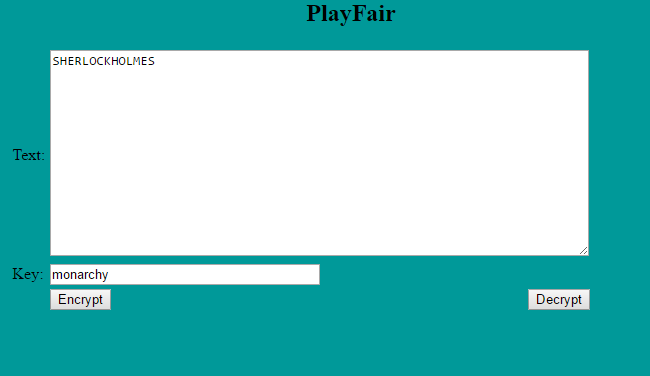
2. Write the plain text that we want to encrypt in the given space labeled “Text”, for example like the one “Sherlock holmes” without space symbol so it would be like “sherlockholmes”. Also input the key need to encrypt the file content, in the example we try “monarchy”



3. After giving both input, click Encrypt button to encrypt the plain text. And will resulting in the following picture.



4. To decrypt it back, give input to the space labeled “Text” with the encrypted text. And the decryption key must be the same as the one being used when encrypting the message. After that, click Decrypt.



5. It would back to the plain text it should be.

## EXPERIMENTAL RESULT

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Input File / Data** | **Key** | **Output File / Data** |
|
| 1. | SherlockHolmes | C = monarchy | PBKMPMDEOVEUIL |
| 2. | Dely Yoga dan Yanto | C = kij | HDOVTGFBFKOXXFOG |
| 3. | Keamanan Informasi dan Jaringan | C = kij | IDINXFXFAMGNMEXNKEXFKIMWOFXF |
| 4. | Informatika ITS | C = informatika | RINFIBMAOMMNKQ |

## REFERENCE(S)

<http://rumkin.com/tools/cipher/playfair.php> (source code)

<http://www.java2s.com/Code/JavaScript/Security/PlayfairEncryptioninJavaScript.htm> (source code)

<https://learncryptography.com/classical-encryption/playfair-cipher> (definition)

## 4. POLYALPHABETIC CIPHER

## DEFINITION

Polyalphabetic cipher is any [cipher](https://en.wikipedia.org/wiki/Cipher) based on [substitution](https://en.wikipedia.org/wiki/Substitution_cipher), using multiple substitution alphabets. One of it’s well known example is Vigenere Cipher, which was adapted as a twist on the standard Caesar cipher to reduce the effectiveness of performing frequency analysis on the ciphertext. The cipher accomplishes this using uses a text string (for example, a word) as a key, which is then used for doing a number of alphabet shifts on the plaintext. Similar to the [Caesar Cipher](https://learncryptography.com/classical-encryption/vigenere-cipher), but instead of performing a single alphabet shift across the entire plaintext, the Vigenère cipher uses a key to determine several different shift amounts across the entirety of the message. But another popular method of this cipher is The Enigma machine which was more complex than any other method, but still use polyalphabetic cipher as it’s basic fundamental.

## DESCRIPTION

In this project, we use javascript to make the encryption and decryption code and combine it with web programming. The input type is text form (read as string) in html and the output type is also in text form.

## SOURCE CODE

File Name : poly.php

Function : doPolyCrypt

Input : isDecrypt (boolean)

Output : textElem (string)

Source code

var vkey = AssignKey();

var vtext= AssignPlain();

// document.write(vtext);

var key = filterKey(vkey);

if (key.length == 0) {

alert("Key has no letters");

return;

}

if (isDecrypt) {

for (var i = 0; i < key.length; i++)

key[i] = (26 - key[i]) % 26;

}

var textElem = vtext;

document.getElementById("plain").value = crypt(textElem, key);

File Name : poly.php

Function : crypt

Input : input (string), key (string)

Output : output (string)

Source code

var output = "";

for (var i = 0, j = 0; i < input.length; i++) {

var c = input.charCodeAt(i);

if (isUppercase(c)) {

output += String.fromCharCode((c - 65 + key[j % key.length]) % 26 + 65);

j++;

} else if (isLowercase(c)) {

output += String.fromCharCode((c - 97 + key[j % key.length]) % 26 + 97);

j++;

} else {

output += input.charAt(i);

}

}

return output;

File Name : poly.php

Function : filterKey

Input : key (string)

Output : result (string)

Source code

var result = [];

for (var i = 0; i < key.length; i++) {

var c = key.charCodeAt(i);

if (isLetter(c))

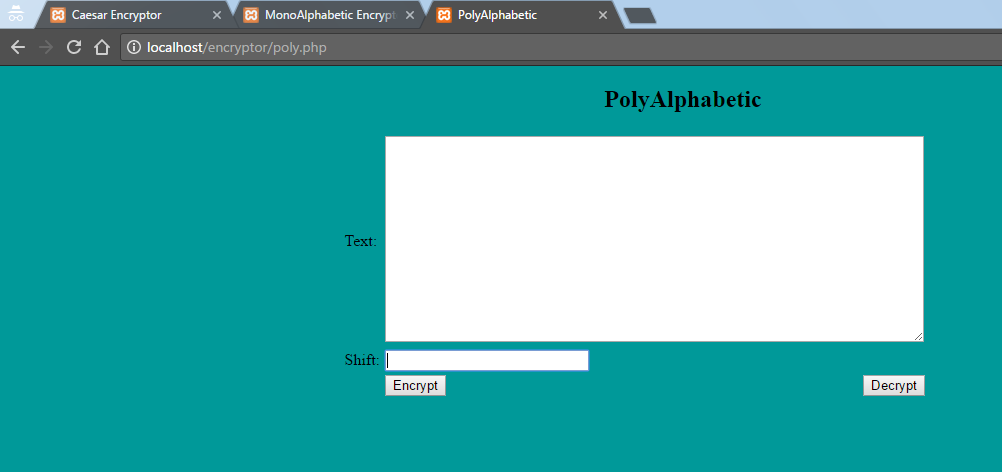
result.push((c - 65) % 32);

}

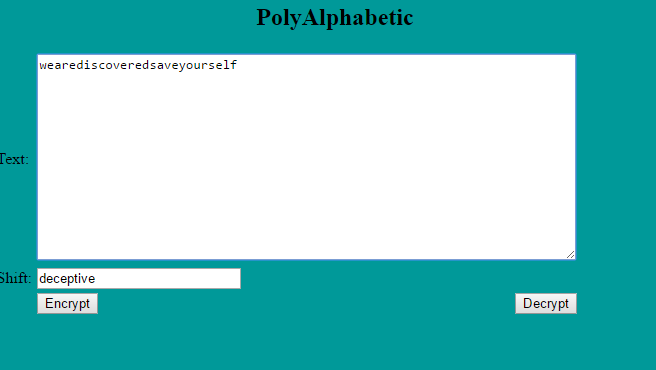
return result;

## IMPLEMENTATION

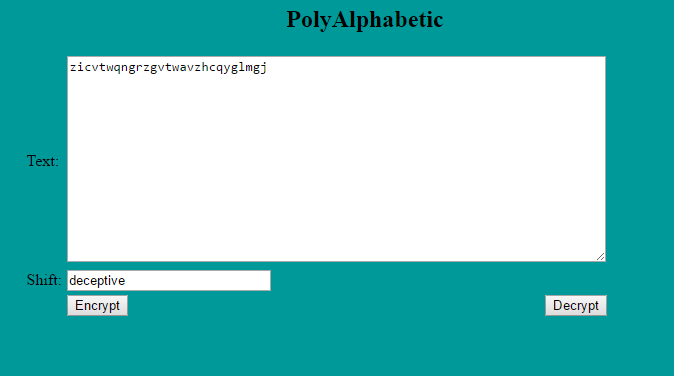
1. Open the php file (localhost/encryption/poly.php)



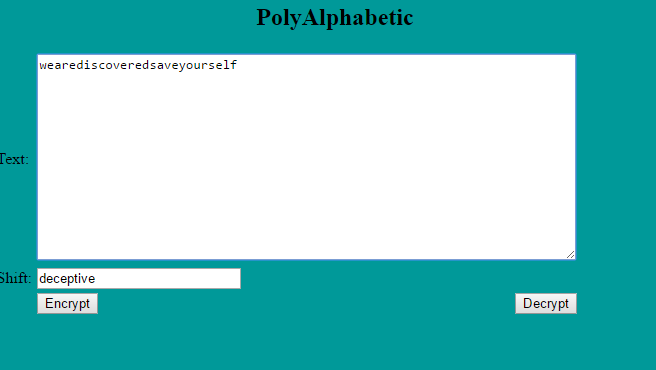
2. Write the plain text that we want to encrypt in the given space labeled “Text”, for example like the one in the pdf book. Also input the encrypt code we want to use in the given space labeled “Shift”, for example we want the encryption key to be “deceptive” which means the 1st letter in plain text is shifted 4(d) alphabet, 2nd is 5(e) alphabet, 3rd is 3(c) alphabet and so on.



3. After giving both input, click Encrypt button to encrypt the plain text. And will resulting in the following picture.



4. To decrypt it back, give input to the space labeled “Text” with the encrypted text. And the decryption key must be the same as the one being used when encrypting the message. After that, click Decrypt.



5. It would back to the plain text it should be.

## EXPERIMENTAL RESULT

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Input File / Data** | **Key** | **Output File / Data** |
|
| 1. | wearediscoveredsaveyourself | C = deceptive | zicvtwqngrzgvtwavzhcqyglmgj |
| 2. | Dely Yoga dan Yanto | C = yantojago | Beyr Mxgg ryn Ltbco |
| 3. | Keamanan Informasi dan Jaringan | C = kij | Umjwiwkv Rxnxbujcq mkv Skzrxojx |
| 4. | Informatika C1F ITS | C = informatika | Qakciyamqua K1S NHJ |

## REFERENCE(S)

<https://en.wikipedia.org/wiki/Polyalphabetic_cipher> (definition)

<https://learncryptography.com/classical-encryption/vigenere-cipher> (definition)

## 5. RAIL FENCE CIPHER

## DEFINITION

Rail Fence Cipher is one example of a transposition cipher. This encryption method use positioning as it’s basic encryption in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, if we have 3 "rails" and a message of 'WE ARE DISCOVERED. FLEE AT ONCE', the cipherer writes out:

W . . . E . . . C . . . R . . . L . . . T . . . E

. E . R . D . S . O . E . E . F . E . A . O . C .

. . A . . . I . . . V . . . D . . . E . . . N . .

Then reads off to get the ciphertext:

WECRLTEERDSOEEFEAOCAIVDEN

## DESCRIPTION

In this project, we use javascript to make the encryption and decryption code and combine it with web programming. The input type is text form (read as string) in html and the output type is also in text form.

## SOURCE CODE

File Name : railfence.php

Function : reilencode

Input :

Output : ctext (string)

Source code

ptext = AssignPlain().toUpperCase().replace(/[^a-zA-Z]/g, '');

key = AssignKey();

mainArray = Array(key);

for (i = 0; i < key; i++) {

mainArray[i] = Array(ptext.length);

for (s = 0; s < ptext.length; s++) {

mainArray[i][s] = '';

}

}

j = 0;

r = 0;

for (i = 0; i < ptext.length; i++) {

p = ptext.substr(i, 1);

mainArray[j][i] = p;

if (r == 0) j = j + 1;

else if (r == 1) j = j - 1;

if (j == key - 1) r = 1;

else if (j == 0) r = 0;

}

for (i = 0; i < mainArray.length; i++) {

mainArray[i] = mainArray[i].join('');

}

ctext = mainArray.join('');

document.getElementById("plain").value = ctext;

File Name : railfence.php

Function : reildecode

Input :

Output : ptext (string)

Source code

ctext = AssignPlain().toUpperCase().replace(/[^a-zA-Z]/g, '');

key = AssignKey();

mainArray = Array(key);

for(i=0; i<key; i++)

{

mainArray[i] = Array(ctext.length);

for (s=0; s<ctext.length; s++)

{

mainArray[i][s] = "";

}

}

q = 0;

for (t=0; t<mainArray.length; t++)

{

j = 0;

r = 0;

for (i=0; i<ctext.length; i++)

{

if (j == t)

{

c = ctext.substr(q,1);

mainArray[j][i] = c;

q = q + 1;

}

if (r == 0) j = j + 1;

else if (r == 1) j = j - 1;

if (j == key - 1) r = 1;

else if (j == 0) r = 0;

}

}

j = 0;

r = 0;

ptext = "";

for (i=0; i<ctext.length; i++)

{

ptext = ptext + mainArray[j][i];

if (r == 0) j = j + 1;

else if (r == 1) j = j - 1;

if (j == key - 1) r = 1;

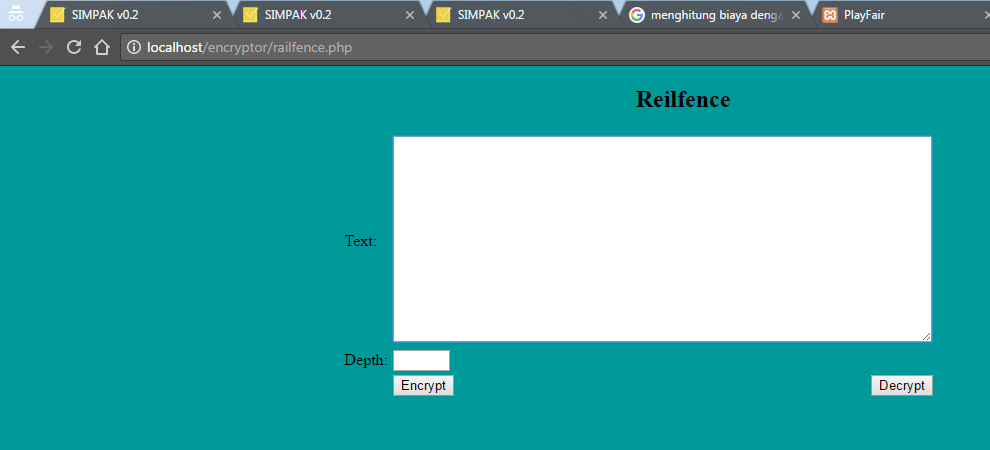
else if (j == 0) r = 0;

}

document.getElementById("plain").value = ptext;

## IMPLEMENTATION

1. Open the php file (localhost/encryption/railfence.php)



2. Write the plain text that we want to encrypt in the given space labeled “Text”, for example “meet me after the toga party”



3. After giving plain text input, click Encrypt button to encrypt the plain text. And will resulting in the following picture.



4. To decrypt it back, give input to the space labeled “Text” with the encrypted text. After that, click Decrypt.



5. It would back to the plain text it should be.

## EXPERIMENTAL RESULT

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Input File / Data** | **Depth** | **Output File / Data** |
| 1. | meetmeafterthetogaparty | 2 | MEMATRHTGPRYETEFETEOAAT |
| 2. | Dely Yoga dan Yanto | 2 | DLYGDNATEYOAAYNO |
| 3. | Keamanan Informasi dan Jaringan | 3 | KAIRIJNEMNNNOMSDNAIGNAAFAARA |
| 4. | Informatika ITS | 5 | IINTKFAAOMISRT |

## REFERENCE(S)

<https://en.wikipedia.org/wiki/Rail_fence_cipher> (definition)

<https://learncryptography.com/classical-encryption/transposition-ciphers> (definition)