

CIFRADO RAIL FENCE



Realizada por:

- **José Francisco Guerrero Collantes.**
- **Rubén Jiménez Ortega.**

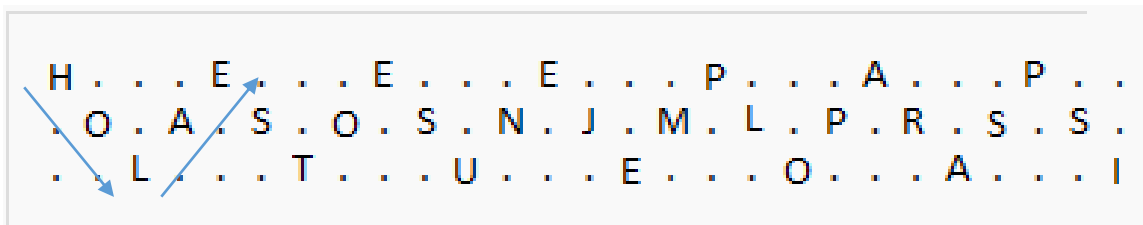
INDICE:

En que consiste.....	3
Problemas.....	3
Cifrado.....	4
Descifrado.....	6
Breve explicación ataque a fuerza bruta.....	8
Ataque a fuerza bruta.....	9

En que consiste:

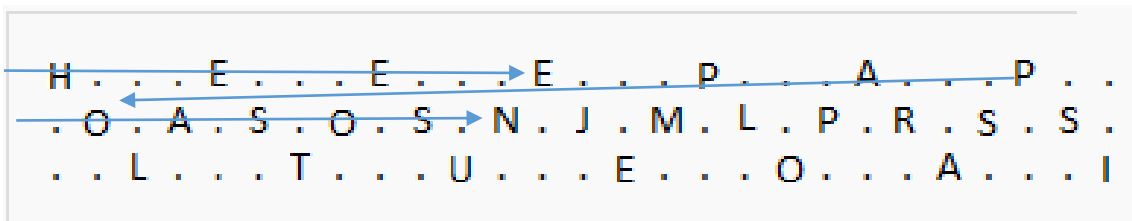
El cifrado Rail Fence es un cifrado por permutación, es decir, que del texto llano, se cambian los caracteres de posición siguiendo un esquema bien definido.

El texto llano se escribe hacia abajo, diagonalmente a través de sucesivos “railes de una valla” y luego hacia arriba, también diagonalmente. Este proceso se repite hasta que se acabe el mensaje que queremos cifrar.



TEXTO PLANO: HOLA ESTO ES UN EJEMPLO PARA PASOS N J M L P R S S I

El cifrado se consigue cogiendo los caracteres de las filas:



TEXTO CIFRADO: HEEPAPOASOSN JMLPRSSLTUEOAI

Problemas:

- Cifrado no muy fuerte.
- Número de claves posibles es pequeño.
- Se podría incluso descifrar a mano.
- La clave es el número de railes.

Cifrado:

Contenido del archivo *Encriptar.py*:

```
def cipher(text, rails):  
    m = (rails - 1) * 2  
    out = ""  
    for i in range(rails):  
        if i % (rails - 1) == 0:  
            out += text[i::m]  
        else:  
            char_pairs = zip(text[i::m], list(text[m-i::m]) + [""])  
            out += ".join(map("".join, char_pairs))  
    return out
```

Contenido del archivo *testRailFence.py*:

```
#!/usr/bin/python

from Encriptar import cipher

print (" ");

print ("Texto sin cifrar: HOLAESTOESUNEJEMPLOPARASPSI");

print (" ");

print ("TEXTO CIFRADO: ");

print cipher("HOLAESTOESUNEJEMPLOPARASPSI", 3);

print (" ");
```

- Introducimos el mensaje que queremos cifrar y el número de railes.
- Cuando ejecutamos, tenemos lo siguiente:

```
ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Encriptar$ python testRailFence.py
Texto sin cifrar: HOLAESTOESUNEJEMPLOPARASPSI
TEXTO CIFRADO:
HEEEPAPOASOSNJMLPRSSLTUEOAI
ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Encriptar$
```

Cuando ejecutamos, si nos fijamos ejecutamos el *testRailFence.py*, que incluye la función *cipher* de *Encriptar.py*, que es donde se hace todo el trabajo de encriptación del mensaje.

Descifrado:

Contenido del archivo *decrypt.py*:

```
def offset(even, rails, rail):  
    if rail == 0 or rail == rails - 1:  
        return (rails - 1) * 2  
    if even:  
        return 2 * rail  
    else:  
        return 2*(rails - 1 - rail)  
  
def decryptRailFence(encrypted, rails):  
    array = [[" " for col in range(len(encrypted))] for row in range(rails)]  
    read = 0  
  
    #Construimos el algoritmo Fence  
    for rail in range(rails):  
        pos = offset(1, rails, rail)  
        even = 0;  
  
        if rail == 0:  
            pos = 0  
        else:  
            pos = int(pos / 2)  
  
        while pos < len(encrypted):  
            if read == len(encrypted):  
                break  
            array[rail][pos] = encrypted[read];  
            read = read + 1
```

```

        pos = pos + offset(even, rails, rail)

        even = not even

    #Devolvemos el mensaje descriptado
    decoded = ""

    for x in range(len(encrypted)):
        for y in range(rails):
            if array[y][x] != " ":
                decoded += array[y][x]

    return decoded

```

Contenido del archivo *testRailFence.py*:

```

#!/usr/bin/python

from decrypt import decryptRailFence

print (" ");

print ("Texto cifrado: HEEPAPOASOSNJMLPRSSLTUEOAI");

print (" ");

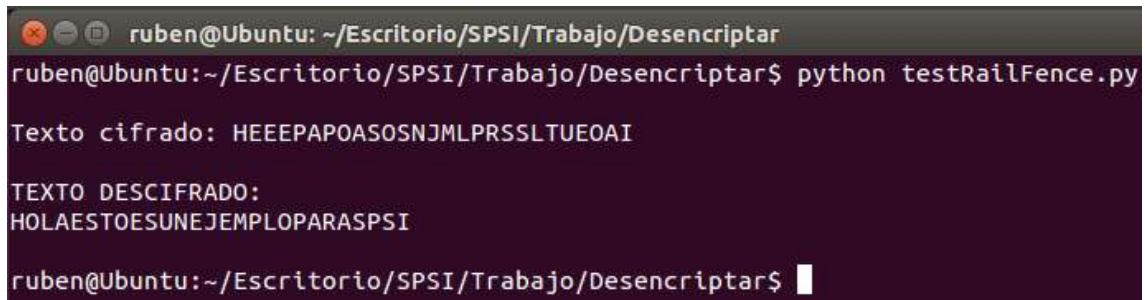
print ("TEXTO DESCIFRADO: ");

print decryptRailFence("HEEPAPOASOSNJMLPRSSLTUEOAI", 3);

print (" ");

```

- Introducimos el mensaje que queremos descifrar y el número de railes.
- Cuando ejecutamos, tenemos lo siguiente:

A terminal window with a dark purple background. The title bar shows 'ruben@Ubuntu: ~/Escritorio/SPSI/Trabajo/Desencriptar'. The prompt is 'ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Desencriptar\$'. The command 'python testRailFence.py' has been executed. The output is: 'Texto cifrado: HEEEPAP0AS0SNJMLPRSSLTUE0AI' followed by a blank line, then 'TEXTO DESCIFRADO:' followed by 'HOLAESTOESUNEJEMPLOPARASPSI'. The prompt is now 'ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Desencriptar\$' with a cursor.

```
ruben@Ubuntu: ~/Escritorio/SPSI/Trabajo/Desencriptar
ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Desencriptar$ python testRailFence.py

Texto cifrado: HEEEPAP0AS0SNJMLPRSSLTUE0AI

TEXTO DESCIFRADO:
HOLAESTOESUNEJEMPLOPARASPSI

ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Desencriptar$
```

Cuando ejecutamos, si nos fijamos ejecutamos el *testRailFence.py*, que incluye la función *decryptRailFence* de *decrypt.py*, que es donde se hace todo el trabajo de descifrado del mensaje.

Breve explicación ataque a fuerza bruta:

- Recorremos un vector desde dos (longitud mínima de la valla) hasta longitud de la cadena + 1.
- Desencriptamos todas las posibles soluciones con el algoritmo anterior para descifrar.
- Imprimimos todos los resultados.

Ataque a fuerza bruta:

Contenido del archivo *FuerzaBruta.py*:

```
from railFence import decryptRailFence

def railBreak(encrypted):
    cipherLen = len(encrypted)
    for i in range(2, cipherLen+1):
        bestGuess = decryptRailFence(encrypted,i,0)
        print(bestGuess)
    return bestGuess
```

Contenido del archivo *testRailFence.py*:

```
#!/usr/bin/python
from FuerzaBruta import railBreak

print (" ");

print ("Texto cifrado: HEEPAPOASOSNJMLPRSSLTUEOAI");

print (" ");

print ("POSIBLES SOLUCIONES:");

print (" ");

print railBreak("HEEPAPOASOSNJMLPRSSLTUEOAI");

print (" ");
```

- Introducimos el mensaje que queremos descifrar.
- Cuando ejecutamos, tenemos lo siguiente:

```

ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Fuerza Bruta$ python testRailFence.py
Texto cifrado: HEEPAPOASOSNJMLPRSSLTUEOAI
POSIBLES SOLUCIONES:
HMELEPERPSASPLOTAUSEOOSANIJ
HOLAESTOESUNEJEMPLOPARASPSI ←
HAMELPEOPORAESSASOESLITNPJU
HPSSOSNAEPJLATMOEALUIEPSEOR
HEAJSOSMSPEAOLLATPSPEONRUIE
HEAJRTAUSMSPEAOLSEIOLPSPEON
HEASJRTAUSMOPEEPOSLSIOLPNA
HEAASMRTAUSLNSPEEP00JPSEIOL
HEAASMRLEIOTSLNSPEEP00JPSUA
HEAASMRLUOIAETSLNSPEEP00JPS
HEAASMPSLUOIAETSLNSPEEP00J
HEAANMPSLUOIAETSLJSSPEEPO
HEPPAONMPSLUOIAETSLJSSOAE
HEEPPAONMPSLUOIAETSLJSSOAE
HEEEPPAONMPSLUOIAETSLJSSOA
HEEEPAPAONMPSLUOIAETSLJSSO
HEEEPAPOAONMPSLUOIAETSLJSS
HEEEPAPOASONMPSLUOIAETSLJS
HEEEPAPOASOSNMPSLUOIAETSLJ
HEEEPAPOASOSNJMPSLUOIAETSL
HEEEPAPOASOSNJMLPSLUOIAETSR
HEEEPAPOASOSNJMLPRSLUOIAETS
HEEEPAPOASOSNJMLPRSSLUOIAET
HEEEPAPOASOSNJMLPRSSLTUEOIAE
HEEEPAPOASOSNJMLPRSSLTUEOIA
HEEEPAPOASOSNJMLPRSSLTUEOAI
HEEEPAPOASOSNJMLPRSSLTUEOAI
ruben@Ubuntu:~/Escritorio/SPSI/Trabajo/Fuerza Bruta$

```

Cuando ejecutamos, si nos fijamos ejecutamos el *testRailFence.py*, que incluye la función *raiBreak* de *FuerzaBruta.py*, y este a su vez llama a la función *decryptRailFence* de *decrypt.py*, y una vez tenemos esto es solo imprimir todos los posibles resultados.

Si nos fijamos en la imagen he señalado la solución de nuestro mensaje. Una vez ejecutado el programa solo queda buscar la solución.