



**udp** UNIVERSIDAD  
DIEGO PORTALES

UNIVERSIDAD DIEGO PORTALES

ESCUELA DE INFORMÁTICA & TELECOMUNICACIONES

---

## Tarea3: Cifrado en producción

---

*Autor:*

*Marcos Fantóval Castro*

*marcos.fantoval@mail.udp.cl*

*19.962.344-3*

*Profesor: Nicolás Boettcher Ayudante: Francisco Lara*

26 de Mayo de 2021

## Índice

1. introducción	2
2. Elección del cifrado	2
3. Explicación Código en Python	2
4. ExplicacionCodigo JavaScript	4
5. Conclusiones	5
6. Link del GitHub	5

## 1. introducción

En el presente informe se realiza un análisis a las distintas formas de implementar el algoritmo 3DES en los lenguajes de programación javascript y python, encriptando un mensaje con python y escondiendo la información de encriptado en un archivo html generado por este script, para luego junto con javascript y tampermonkey desencriptar este mensaje.

## 2. Elección del cifrado

Tras una búsqueda rigurosa, se elige el algoritmo 3DES (Triple DES osea aplica tres veces el encriptado Data Encryption Standard)

Su librería en Python es des y en JavaScript es crypto-js.

En este informe se usará el siguiente ejemplo para comprobar el correcto funcionamiento de los códigos.

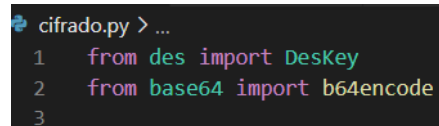
-Palabra a utilizar: 'tarea numero 3'

-Llave a utilizar: 'DESllave'

-Mensaje cifrado: 'w4vyqS9R1nSzc8kZnT6TDQ=='

## 3. Explicación Código en Python

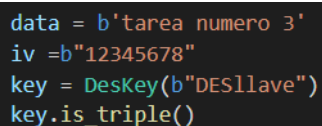
En primer lugar, se implementa la librería DesKey para obtener todas sus funciones y poder ejecutar de forma correcta el código:



```
❏ cifrado.py > ...  
1  from des import DesKey  
2  from base64 import b64encode  
3
```

Figura 1: Se importan las librerías

Luego se declaran las variables, las cuales son: key (texto el cual es necesario para descifrar de forma correcta un mensaje encriptado), el texto en sí como data (el mensaje el cual se desea cifrar), el modo de operación (en este caso CBC) se ve en un IV (iniciador de vector, el cual es el valor inicial utilizado para iniciar el proceso de encriptación). Para encriptar el mensaje, simple-



```
data = b'tarea numero 3'  
iv = b"12345678"  
key = DesKey(b"DESllave")  
key.is_triple()
```

Figura 2: Se le da valor a las variables

mente se utiliza la función `encrypt` que otorga la librería ingresando todos los parámetros necesarios incluyendo `padding` igual a `'True'`, esto lo que hace es rellenar cuando falten caracteres. Es importante mencionar que todo el proceso se ejecuta en binario (por esto en cada variable hay una letra `b` al comienzo) por lo que para pasar de binario a Base64 se utiliza una función que viene incluida dentro de la librería `base64`.

```

9  encryption = key.encrypt(data, initial=iv, padding=True)
10 encryption = b64encode(encryption).decode('utf-8')
11 print(encryption)

```

Figura 3: Se genera el encriptado del mensaje

Una vez realizado el proceso se corrobora y se imprime en el terminal el resultado, como se observa a continuación:

```

10 encryption = b64encode(encryption)
11 print(encryption)
12
13 html = """<p>Este sitio contiene un
14 <div class='tripleDES' id=''+enc
15 <script src="https://cdnjs.cloudflare
16 <script src="./tamper.js"></script>
17
18 archivo = open("pagina.html", "w")
19 archivo.write(html)
20 archivo.close()

```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE

```

Marcos F@LAPTOP-6U6AS940 MINGW64 ~/Desktop/
$ "C:/Users/Marcos F/AppData/Local/Programs
w4vyqS9R1nSzc8kZnT6TDQ==

```

Figura 4: Se muestra el el mensaje encriptado

Luego, para generar el archivo HTML donde estará incluido el mensaje encriptado, se utiliza el siguiente código:

```

html = """<p>Este sitio contiene un mensaje secreto</p>
<div class='tripleDES' id=''+encryption+''></div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js">
<script src="./tamper.js"></script>"""

archivo = open("pagina.html", "w")
archivo.write(html)
archivo.close()

```

Figura 5: Código de creación y generación del HTML

El código anterior crea el archivo e imprime el HTML generado, en donde el mensaje encriptado va incluido en el `id` del `div` del código y se le inserta las librerías necesarias para el código JS como adición al tampermonkey.

## 4. Explicacion Codigo JavaScript

Al inicio del codigo se presentan las configuraciones dentro de estas se indica el link del html a utilizar el script también se indica la librería a utilizar en este caso crypto-js.

```
// ==UserScript==
// @name      3DES Decrypt
// @namespace  http://tampermonkey.net/
// @version    2.0
// @description Descriptar un codigo cifrado
// @author     Marcos Fantóval C.
// @match      https://marcosfantoal2.github.io/CriptoT3/
// @grant      none
// @require    https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js
// ==/UserScript==
```

Figura 6: "Configuración" del Tampermonkey

Luego se declaran las variables que de necesitan en la función de desenscriptado en este caso la llave y el iv.

```
var iv = "12345678";
var key = "DESllave";
function decryptByDES(ciphertext)
```

Figura 7: Creación de variables

Se crea la función que va a desenscriptar el mensaje esta recibe un mensaje y las dos variables creadas anteriormente. Dentro de esta se crea la variable keyh en hexadecimal con el parámetro key y lo mismo para el ivh, ambas con base utf-8. A continuación ya con las variables creadas se utiliza la función decrypted con los parámetros del mensaje la keyh,ivh y se le agrega el modo en este caso CBC junto con el padding. Para finalizar la funcion retorna un mensaje que corresponde al mensaje cifrado pero en texto plano.

. Se crea una variable 'div' en donde se guarda el elemento 'div' que se obtiene del html y que

```
function decryptByDES(ciphertext, key, iv) {
  var keyH = CryptoJS.enc.Utf8.parse(key);
  var ivH = CryptoJS.enc.Hex.parse(CryptoJS.enc.Utf8.parse(iv).toString(CryptoJS.enc.Hex));
  var decrypted = CryptoJS.TripleDES.decrypt(ciphertext, keyH, { iv: ivH, mode: CryptoJS.mode.CBC, padding: CryptoJS.pad.Pkcs7 });
  return decrypted;
}
```

Figura 8: Función de desenscriptado

contiene el mensaje cifrado en el id. Luego se el id con la finalidad de desencriptarlo, posteriormente se llama a la función creada anteriormente. Para finalmente pasar lo obtenido en la función a texto plano.

```
var div = document.getElementsByTagName("div");  
var idTextoCifrado = div[0].id;  
var mensaje_tx_plano = decryptByDES(idTextoCifrado, key, iv);  
mensaje_tx_plano = mensaje_tx_plano.toString(CryptoJS.enc.Utf8);
```

Figura 9: Encontrar y descifrar el mensaje

Por ultimo se agrega el descifrado a la etiqueta del div y también se muestra el mensaje por consola teniendo ambas partes para ver el mensaje.

```
div[0].innerHTML = 'mensaje cifrado: '+mensaje_tx_plano;  
console.log(mensaje_tx_plano.toString(CryptoJS.enc.Utf8))
```

Figura 10: Mostrar el mensaje en el HTML y por consola

## 5. Conclusiones

Aunque fuera complicado encontrar las librerías de ambos lenguajes y que fueran compatibles en la implementación del algoritmo, se logro realizar el encriptado en python y el desencriptado el el script. Por otro lado para tampermonkey, hubo uno que otro problema para integrar la librería de javascript, aunque finalmente se lograra llevar acabo el cometido de la tarea.

Ya tampermonkey se a tenido que utilizar durante el semestre por ejemplo para ver las pautas se puede decir que no es una cosa totalmente desconocida pero no se había aprovechado al máximo de hecho, es bastante útil para automatizaciones que necesiten ser implementadas a alguna pagina. Debido al tiempo de búsqueda que se tuvo que implementar para encontrar las librerías se puede concluir que si bien existen estas, no necesariamente están preparadas para ser compatibles con las de otros lenguajes, osea pensando que ya es difícil que sean compatibles entre si aun mas con otros lenguajes aunque como ya se demostró no es imposible.

## 6. Link del GitHub

<https://github.com/MarcosFantoal2/CriptoT3>