

# Informe Laboratorio 4

## Sección 1

Alumno: Philip Andre Magna Depoortere  
e-mail: philip.magna\_d@mail.udp.cl

Noviembre de 2023

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo (Parte 1)</b>	<b>3</b>
2.1. Detecta el cifrado utilizado por el informante . . . . .	3
2.2. Logra que el script solo se gatille en el sitio usado por el informante . . . . .	4
2.3. Define función que obtiene automáticamente el password del documento . . . . .	5
2.4. Muestra la llave por consola . . . . .	5
<b>3. Desarrollo (Parte 2)</b>	<b>6</b>
3.1. reconoce automáticamente la cantidad de mensajes cifrados . . . . .	6
3.2. muestra la cantidad de mensajes por consola . . . . .	6
<b>4. Desarrollo (Parte 3)</b>	<b>7</b>
4.1. Importa la librería cryptoJS . . . . .	7
4.2. Utiliza SRI en la librería CryptoJS . . . . .	7
4.3. Logra decifrar uno de los mensajes . . . . .	8
4.4. Imprime todos los mensajes por consola . . . . .	9
4.5. Muestra los mensajes en texto plano en el sitio web . . . . .	9
4.6. El script logra funcionar con otro texto y otra cantidad de mensajes . . . . .	10
4.7. Indica url al código .js implementado para su validación . . . . .	11

## 1. Descripción de actividades

Para este laboratorio, deberá utilizar Tampermonkey y la librería CryptoJS (con SRI) para lograr obtener los mensajes que le está comunicando su informante. En esta ocasión, su informante fue más osado y se comunicó con usted a través de un sitio web abierto a todo el público <https://cripto.tiiny.site/>.

Sólo un ojo entrenado como el suyo logrará descifrar cuál es el algoritmo de cifrado utilizado y cuál es la contraseña utilizada para lograr obtener la información que está oculta.

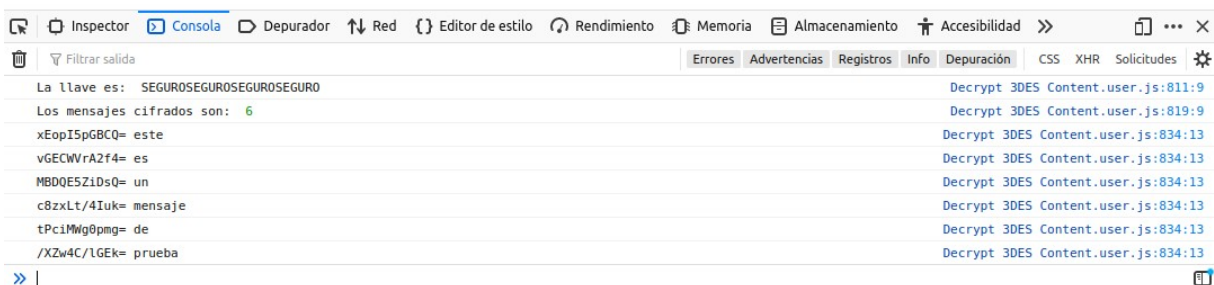
1. Desarrolle un plugin para tampermonkey que permita obtener la llave para el descifrado de los mensajes ocultos en la página web. La llave debe ser impresa por la consola de su navegador al momento de cargar el sitio web. Utilizar la siguiente estructura:
  - La llave es: KEY
2. En el mismo plugin, se debe detectar el patrón que permite identificar la cantidad de mensajes cifrados. Debe imprimir por la consola la cantidad de mensajes cifrados. Utilizar la siguiente estructura: Los mensajes cifrados son: NUMBER
3. En el mismo plugin debe obtener cada mensaje cifrado y descifrarlo. Ambos mensajes deben ser informados por la consola (cifrado espacio descifrado) y además cada mensaje en texto plano debe ser impreso en la página web.

El script desarrollado debe ser capaz de obtener toda la información del sitio web (llave, cantidad de mensajes, mensajes cifrados) sin ningún valor forzado. Para verificar el correcto funcionamiento de su script se utilizará un sitio web con otro texto y una cantidad distinta de mensajes cifrados. Deberá indicar la url donde se podrá descargar su script.

Un ejemplo de lo que se debe visualizar en la consola, al ejecutar automáticamente el script, es lo siguiente:

Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica. Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica. Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica.

este  
es  
un  
mensaje  
de  
prueba



## 2. Desarrollo (Parte 1)

Primeramente, se lleva a cabo la creación de un plugin para **Tampermonkey**, que permita obtener la llave para el descifrado de los mensajes ocultos en la página web <https://cripto.tiiny.site>.

### 2.1. Detecta el cifrado utilizado por el informante

Primero, se busca detectar el tipo de cifrado que se utiliza en el sitio web.

## 2.2 Logra que el script solo se gatille en el sitio usado por el informante

```
const descifrarMensaje = (mensajeEncriptado, claveSeguridad) => {  
  const claveFormateada = CryptoJS.enc.Utf8.parse(claveSeguridad);  
  const opcionesDescifrado = { mode: CryptoJS.mode.ECB, padding: CryptoJS.pad.Pkcs7 };  
  const textoCifrado = CryptoJS.enc.Base64.parse(mensajeEncriptado);  
  const descifrado = CryptoJS.TripleDES.decrypt({ ciphertext: textoCifrado }, claveFormateada, opcionesDescifrado);  
  return descifrado.toString(CryptoJS.enc.Utf8);  
};
```

Figura 1: Detección del Algoritmo de Cifrado que se utiliza.

Como se aprecia en la figura 1, se tiene que el algoritmo de Cifrado **Triple DES** o **3DES** corresponde al que utiliza el informante para entregar los mensajes cifrados. En él, se plantea una mejora en la seguridad cifrando tres veces con tres claves diferentes o con la misma clave en tres rondas de forma sucesiva. El método **.decrypt** se refiere a la realización de la operación de descifrado, en donde se le pasa el texto cifrado (**ciphertext: textoCifrado**) como parte de un objeto con la propiedad. La clave **claveFormateada** corresponde a la key formateada para el proceso de descifrado, además de que se pasan las operaciones de descifrado (**opcionesDescifrado**), que incluyen el modo de operación y el relleno respectivo.

## 2.2. Logra que el script solo se gatille en el sitio usado por el informante

Luego, se muestra un trozo de un código en JavaScript que se gatilla solamente en el sitio web que utiliza el informante.

```
// ==UserScript==  
// @name      Laboratorio 4 - Criptografia y Seguridad en Redes  
// @namespace  http://tampermonkey.net/  
// @version   1.0  
// @description LAB 4 CRIPTO  
// @author    Philip Magna  
// @match     https://cripto.tiiny.site/  
// @require   https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js  
// @icon      https://cdn.iconscout.com/icon/free/png-256/free-monkey-67-450339.png  
// @grant     none  
// @run-at    document-end  
// ==/UserScript==
```

Figura 2: Match hacia el sitio web que se utiliza.

Como se puede apreciar en la figura 2, se tiene un script hecho con **JavaScript**, específicamente, tal como se muestra, la directiva **@match** define el ámbito de la página donde se aplica el código, pues con la URL **https://cripto.tiiny.site/** especifica que el script debe actuar únicamente en tal sitio.

## 2.3. Define función que obtiene automáticamente el password del documento

A continuación, se presenta una función para el script que obtenga de forma automática la contraseña (**password**) del documento del sitio web.

```
const extraerInicialesMayusculas = () => {
  const textoCompleto = document.body.innerText;
  return textoCompleto.split('.')
    .map(frase => frase.trim().charAt(0))
    .filter(caracterInicial => caracterInicial !== '')
    .join('');
};
```

Figura 3: Función para obtener de forma automática la llave (password) del documento.

Tal como se muestra en la figura 3, se define una función para procesar el texto que se presenta en la página web, para así extraer las iniciales mayúsculas de cada oración. Adicionalmente, se declara una constante llamada **extraerInicialesMayusculas**, que almacena una función flecha, pues dicha función no admite argumentos y entrega el resultado de la expresión correspondiente. Como resultado de la ejecución del código, corresponde a una cadena de texto que contiene la primera letra de cada oración que se presenta en el cuerpo del documento, siempre y cuando sea primera letra y no sea un espacio en blanco, lo que se utiliza para obtener una clave en base al texto.

## 2.4. Muestra la llave por consola

Ahora, en base al punto anterior, se lleva a cabo la impresión de la llave obtenida a través de la consola del navegador (**browser**).

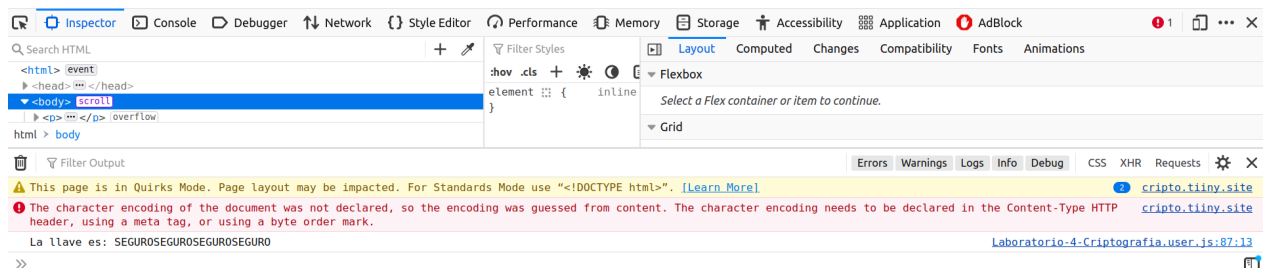


Figura 4: Muestreo de la llave (KEY) por consola del navegador web.

En la figura 4, dentro de la consola del navegador se muestra el resultado directo de la ejecución del script analizado anteriormente. Tal mensaje resulta significativo, ya que representa el funcionamiento de la función **extraerInicialesMayusculas**. La llave sugiere que el texto proporcionado en el sitio web contiene la palabra **SEGURO** cuatro veces, pues en él se toma en cuenta solo las mayúsculas.

### 3. Desarrollo (Parte 2)

#### 3.1. reconoce automáticamente la cantidad de mensajes cifrados

A continuación, se presenta el procedimiento para obtener, de forma automática, la cantidad de mensajes cifrados del sitio web, en base a la información anterior.

```
const buscarDivsConFormatoEspecifico = (clave) => {
  const divs = [...document.getElementsByTagName('div')];
  const patronFormato = /^M\d+$/;

  return divs.reduce((resultados, divActual) => {
    if (patronFormato.test(divActual.classList[0])) {
      const mensajeDescifrado = descifrarMensaje(divActual.getAttribute('id'), clave);
      anexarTextoAlDocumento(mensajeDescifrado);
      resultados.push(divActual.getAttribute('id') + ' ' + mensajeDescifrado);
    }
    return resultados;
  }, []);
};

const descifrarMensaje = (mensajeEncriptado, claveSeguridad) => {
  const claveFormateada = CryptoJS.enc.Utf8.parse(claveSeguridad);
  const opcionesDescifrado = { mode: CryptoJS.mode.ECB, padding: CryptoJS.pad.Pkcs7 };
  const textoCifrado = CryptoJS.enc.Base64.parse(mensajeEncriptado);
  const descifrado = CryptoJS.TripleDES.decrypt({ ciphertext: textoCifrado }, claveFormateada, opcionesDescifrado);
  return descifrado.toString(CryptoJS.enc.Utf8);
};

const anexarTextoAlDocumento = (texto) => {
  const nuevoDiv = document.createElement("div");
  nuevoDiv.textContent = texto;
  document.body.appendChild(nuevoDiv);
};
```

Figura 5: Reconocimiento automático de la cantidad de mensajes cifrados.

Tal como se aprecia en la figura 5, se elabora un código en JavaScript en Tampermonkey, con el fin de obtener la cantidad correspondiente de los mensajes cifrados. Específicamente, primero se seleccionan todos los elementos **div** del documento y se almacenan en un arreglo llamado **divs**, considerando la definición de una expresión regular **patronFormato**, para poder identificar los **divs** con una clase que sigue un patrón específico. En adición, se utiliza otra función llamada **anexarTextoalDocumento**, en donde básicamente se crea un nuevo elemento **div** en el documento (se asigna el **texto** proporcionado como el contenido del nuevo **div**) y añade un nuevo **div** al final del cuerpo de lo anterior.

#### 3.2. muestra la cantidad de mensajes por consola

En base al punto anterior, se lleva a cabo el muestreo de la cantidad de mensajes a través de la consola del navegador web.

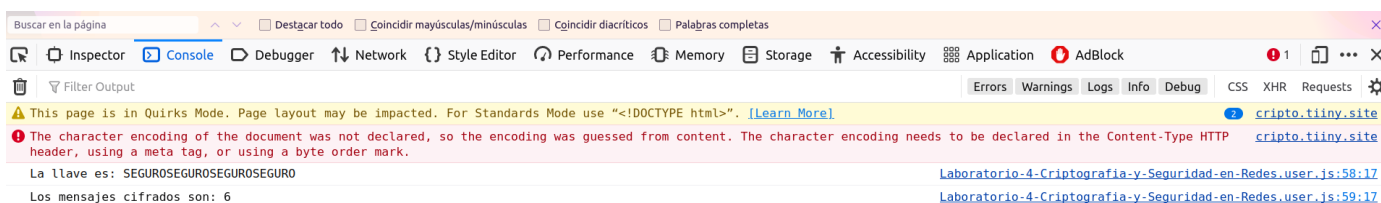


Figura 6: Muestreo de la cantidad de mensajes por consola del navegador.

En la figura 6, se muestra el número de mensajes que se cifran en la consola del navegador web. La función **buscarDivsConFormatoEspecifico** busca en el documento todos los ejemplos **div** que coincidan con un patrón de clase específico, como se indica en la expresión regular anteriormente mencionada, pues cada **div** que coincida con tal patrón tiene su **ID** descifrado, utilizando la función **descifrarMensaje**, pues de esa manera, en la consola se indica que se encuentran y procesan seis divs que cumplen con el criterio de búsqueda.

## 4. Desarrollo (Parte 3)

### 4.1. Importa la librería cryptoJS

A continuación, se lleva a cabo el proceso de importar la librería **CryptoJS**, considerando la información anterior que se menciona.

```
// @match *
// @require https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js
```

Figura 7: Importación de la Librería CryptoJS.

Como se muestra en la figura 7, la línea con el dato **@match** indica que el script depende de la librería **CryptoJS**, junto con la versión específica que se importa, la cual viene siendo la **4.0.0**. Además, la directiva **@require** asegura que antes de ejecutar el código, la extensión **Tampermonkey** se encarga de cargar la librería **CryptoJS** desde la dirección URL especificada anteriormente, lo que permite que el script pueda utilizar diversas funciones de la librería respectiva.

### 4.2. Utiliza SRI en la librería CryptoJS

Luego, se muestra el uso de **SRI** en la librería **CryptoJS**.



```
// Función para cargar CryptoJS con SRI
function cargarCryptoJS() {
    return new Promise((resolve, reject) => {
        const script = document.createElement('script');
        script.setAttribute('src', cryptoJsUrl);
        script.setAttribute('integrity', sriHash);
        script.setAttribute('crossorigin', 'anonymous');
        script.onload = resolve;
        script.onerror = reject;
        document.head.appendChild(script);
    });
}

// Ejecutar el resto del código una vez que CryptoJS se haya cargado
cargarCryptoJS().then(() => {
    const extraerInicialesMayusculas = () => {
        const textoCompleto = document.body.innerText;
        return textoCompleto.split('.')
            .map(frase => frase.trim().charAt(0))
            .filter(caracterInicial => caracterInicial !== '')
            .join('');
    };
});
```

Figura 8: Uso de SRI en la librería CryptoJS.

Tal como se muestra en la figura 8, se lleva a cabo el procedimiento de cargar la librería **CryptoJS** con **Subresource Integrity (SRI)**, además de definir las funciones que procesan el texto en la página web. Se define una función llamada **cargarCryptoJS**, que retorna una **Promise**, que viene siendo una promesa para realizar las operaciones asíncronas que se piden, lo que permite un flujo de control más claro, especialmente cuando se encadenan diversos procesos. Dentro del elemento **script** se establecen varios atributos, los cuales son una URL de la librería CryptoJS que se carga (**src**), un hash del SRI (**integrity**), que debe coincidir con el contenido del archivo en la URL que se plantea.

### 4.3. Logra decifrar uno de los mensajes

Luego de realizar los dos procesos anteriores, se procede a hacer el proceso de descifrar uno de los mensajes que se tienen en el sitio web.

xEopI5pGBCQ= este

[Lab4-Cripto.user.js:79:53](#)

Figura 9: Descifrado de uno de los mensajes.



Como se puede apreciar en la figura 9, se muestra una cadena de texto que viene siendo uno de los divs del sitio web descifrado. Específicamente, la cadena cifrada **xEopI5pGBCQ=**, al descifrarla, se obtiene el mensaje **este**, pues se asocia con el **id** de dicho elemento **div** de la página web.

#### 4.4. Imprime todos los mensajes por consola

Ahora, considerando la información anterior, se imprimen todos los mensajes a través de la consola del navegador web.

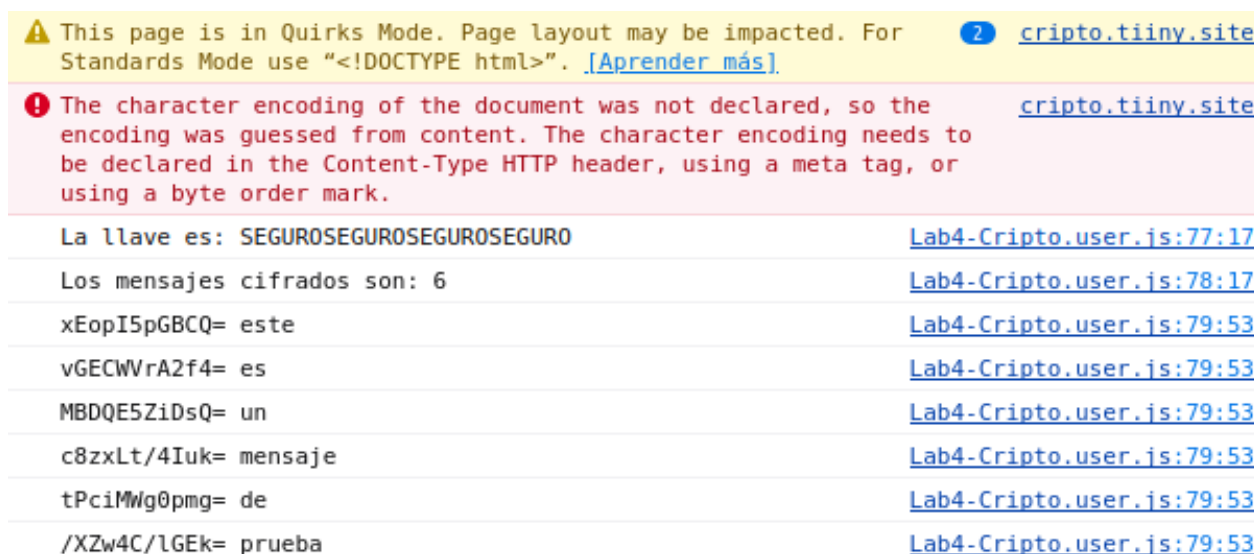


Figura 10: Impresión de todos los mensajes por consola.

Como se puede observar en la figura 10, se listan las cadenas de texto, que vienen siendo los mensajes descifrados correspondientes a los mensajes cifrados mencionados anteriormente. Cada cadena descifrada aparece junto a su respectivo texto plano.

#### 4.5. Muestra los mensajes en texto plano en el sitio web

Luego, se muestran los mensajes en texto plano en el sitio web.

copiada, modificada o eliminada. Otras  
tareas de las que pueden ser  
responsables los criptoanalistas  
incluyen evaluar, analizar y localizar las  
debilidades en los sistemas y  
algoritmos de seguridad criptográfica.

este  
es  
un  
mensaje  
de  
prueba

Figura 11: Muestreo de los mensajes en texto plano en el sitio web.

Como se muestra en la figura 11, se tienen los mensajes que se ven en la consola del navegador. Pues se tiene que cada cadena codificada, una vez que se descifra, revela la palabra respectiva, que cuando se unen forman la oración **este es un mensaje de prueba**, específicamente, se identifican los elementos **div** en la página web, que tienen un formato especial en el atributo **class** o **id**, además de utilizar la llave de descifrado (**SEGUROSEGUROSEGURO**).

#### 4.6. El script logra funcionar con otro texto y otra cantidad de mensajes

A continuación, se lleva a cabo el procedimiento de usar el script elaborado considerando otro texto y otra cantidad de mensajes.

S. E. G. U. R. O. S. E. G. U. R. O. S. E. G. U. R. O. S. E.  
G. U. R. O.

hola  
soy  
un  
informatico

Figura 12: Demostración del funcionamiento del código con otro texto y otra cantidad de mensajes I.

#### 4.7 Indica url al código .js implementado para su validación# DESARROLLO (PARTE 3)

Como se puede apreciar en la figura 12, se muestra una demostración del funcionamiento del código creado anteriormente, considerando otro texto a utilizar con una llave adjunta, para así descifrar otra cantidad de mensajes, pues se utiliza el algoritmo de Cifrado **Triple DES** o **3DES** para llevar a cabo lo mencionado.

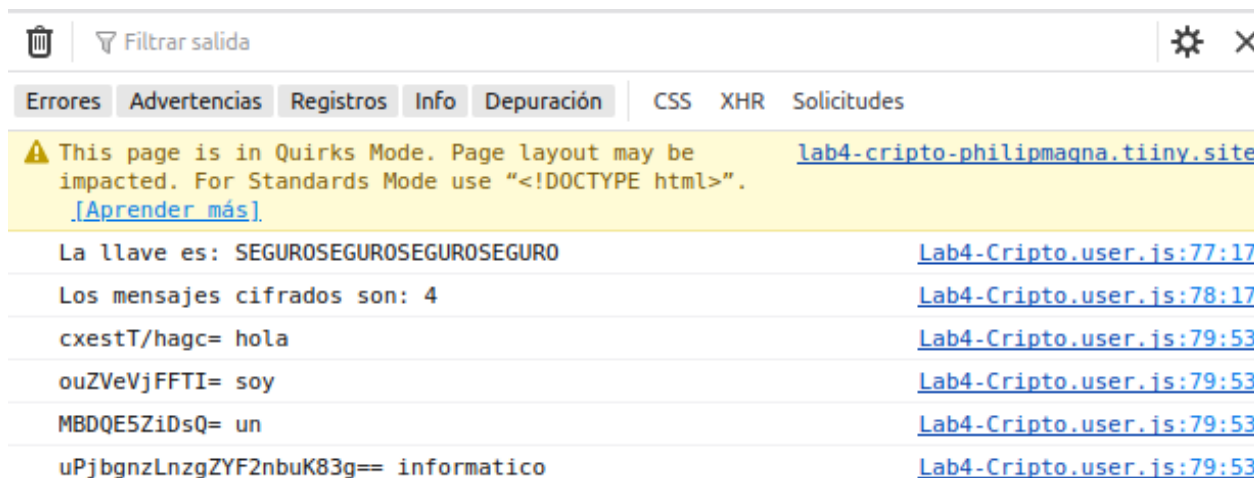


Figura 13: Demostración del funcionamiento del código con otro texto y otra cantidad de mensajes II.

Después, se tiene la impresión de llave obtenida en base al texto anterior, junto con la cantidad de mensajes cifrados. Cada mensaje se muestra en un lado izquierdo en su modo **cifrado** y en el derecho **descifrado**.


#### 4.7. Indica url al código .js implementado para su validación


Por último, se indica la URL para obtener el código de JavaScript que se implementa, con el fin de realizar su validación en diversas pruebas.


**URL:** <https://rb.gy/xws84a>

**URL Alternativa:** <https://easyupload.io/vike77>

## 4.7 Indica url al código .js implementado para su validación DESARROLLO (PARTE 3)

Lab-4-Criptografia / Lab4-Cripto-PhilipMagna.js 

 **MagnumPineapple** Add files via upload

Code Blame 86 lines (74 loc) · 3.38 KB  Code 55% faster with GitHub Copilot

```
1  // ==UserScript==
2  // @name      Lab4-Cripto
3  // @namespace  http://tampermonkey.net/
4  // @version   1.0
5  // @description Laboratorio N°4 de Criptografia y Seguridad en Redes
6  // @author    Philip Magna
7  // @match     *
8  // @require   https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js
9  // @icon      https://cdn.iconscout.com/icon/free/png-256/free-monkey-67-450339.png
10 // @grant     none
11 // @run-at    document-end
12 // ==/UserScript==
13
14
15 (function() {
16     'use strict';
17
18     // La URL de la libreria CryptoJS
19     const cryptoJsUrl = 'https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js';
20     const sriHash = 'sha384-0DrKBsfUuJe/vqjia1HviapRn4mR1BYfCpQ9gT7qjSKu8TrzTe2tlbK3cI9i9EwV';
21
22     // Función para cargar CryptoJS con SRI
23     function cargarCryptoJS() {
24         return new Promise((resolve, reject) => {
25             const script = document.createElement('script');
26             script.setAttribute('src', cryptoJsUrl);
27             script.setAttribute('integrity', sriHash);
28             script.setAttribute('crossorigin', 'anonymous');
29             script.onload = resolve;
30             script.onerror = reject;
31             document.head.appendChild(script);
```

Figura 14: URL de descarga del Código de JavaScript utilizado I.

En la figura 14, se tiene el sitio web que se obtiene como resultado del uso de la URL para descargar el código de JavaScript elaborado.

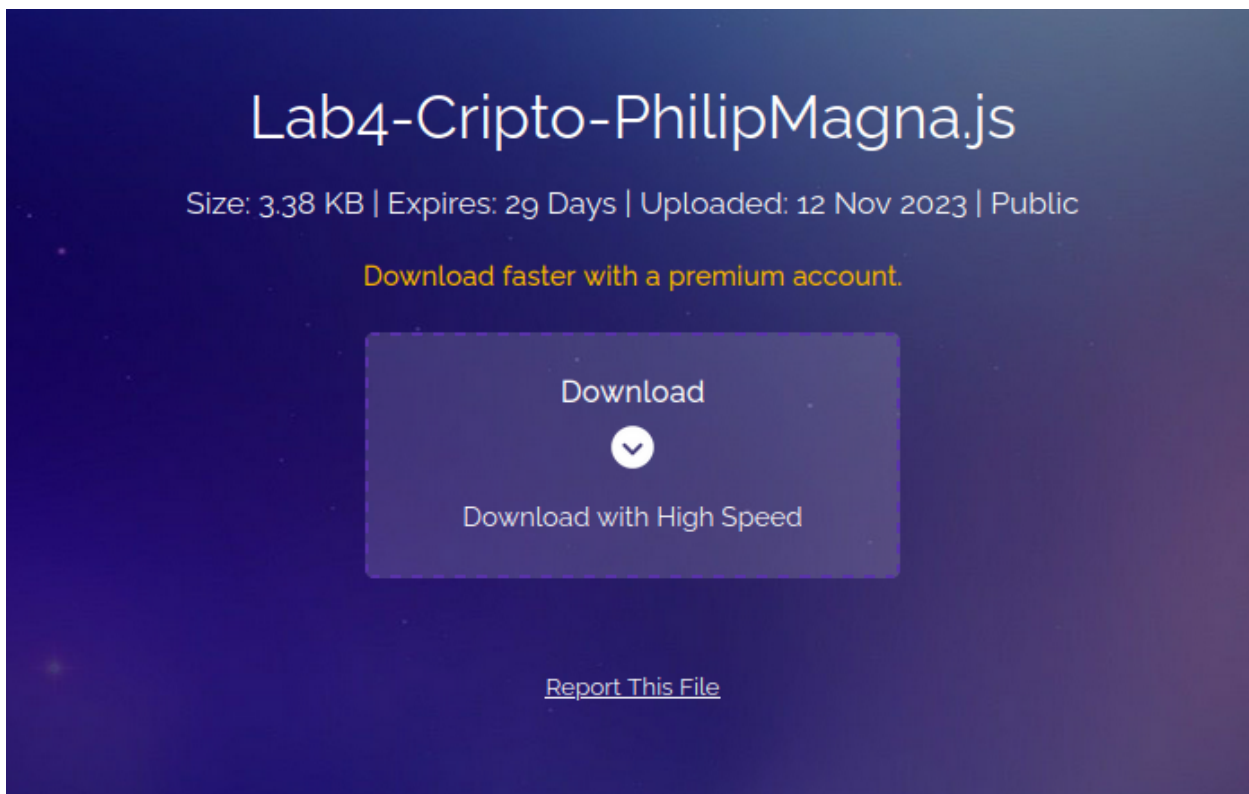


Figura 15: URL de descarga del Código de JavaScript utilizado II.

Finalmente, en la figura 15, se tiene otra página web para descargar el archivo con el código **.js**, en base un link alternativo o secundario.

## Conclusiones y comentarios

Se concluye que la presente experiencia ha permitido profundizar, en la aplicación práctica y teórica, los fundamentos criptográficos y sus usos en la seguridad de la información. A través de diversas actividades realizadas, se ha conseguido una comprensión más clara de cómo los algoritmos de cifrado pueden proteger la información contra el acceso no autorizado y cómo la criptoanálisis resulta fundamental para evaluar la robustez de tales sistemas criptográficos.

Adicionalmente, se ejecutaron una serie de ejercicios de cifrado y descifrado que destacan la importancia de utilizar claves de longitud adecuada, además de la implementación de buenas prácticas en la generación de claves seguras. La experimentación con diferentes tamaños de claves refuerza el concepto de que claves más largas y complejas ofrecen una mayor seguridad, que viene siendo una buena práctica.

También, la observación de la salida de los mensajes cifrados y descifrados demuestra la efectividad de los métodos criptográficos empleados, así como la necesidad de comprender

completamente las herramientas y técnicas utilizadas para asegurar la confidencialidad y la integridad de la información.

Se puede decir que, al haber realizado las tareas planteadas, se ha obtenido una mayor comprensión de los temas tratados, lo que resulta muy importante implementar buenas prácticas dentro del área de la criptografía y seguridad en redes.