

# Informe Laboratorio 4

## Sección 2

Alejandro Arratia

e-mail: alejandro.arratia@mail.udp.cl

Junio de 2024

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo (Parte 1)</b>	<b>4</b>
2.1. Detecta el cifrado utilizado por el informante . . . . .	4
2.2. Logra que el script solo se gatille en el sitio usado por el informante . . . . .	5
2.3. Define función que obtiene automáticamente el password del documento . . .	6
2.4. Muestra la llave por consola . . . . .	6
<b>3. Desarrollo (Parte 2)</b>	<b>7</b>
3.1. Reconoce automáticamente la cantidad de mensajes cifrados . . . . .	7
3.2. Muestra la cantidad de mensajes por consola . . . . .	7
<b>4. Desarrollo (Parte 3)</b>	<b>8</b>
4.1. Importa la librería cryptoJS . . . . .	8
4.2. Utiliza SRI en la librería CryptoJS . . . . .	9
4.3. Repercusiones de SRI inválido . . . . .	9
4.4. Logra descifrar uno de los mensajes . . . . .	10
4.5. Imprime todos los mensajes por consola . . . . .	11
4.6. Muestra los mensajes en texto plano en el sitio web . . . . .	12
4.7. El script logra funcionar con otro texto y otra cantidad de mensajes . . . . .	13
4.8. Indica url al código .js implementado para su validación . . . . .	16

## 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/>.

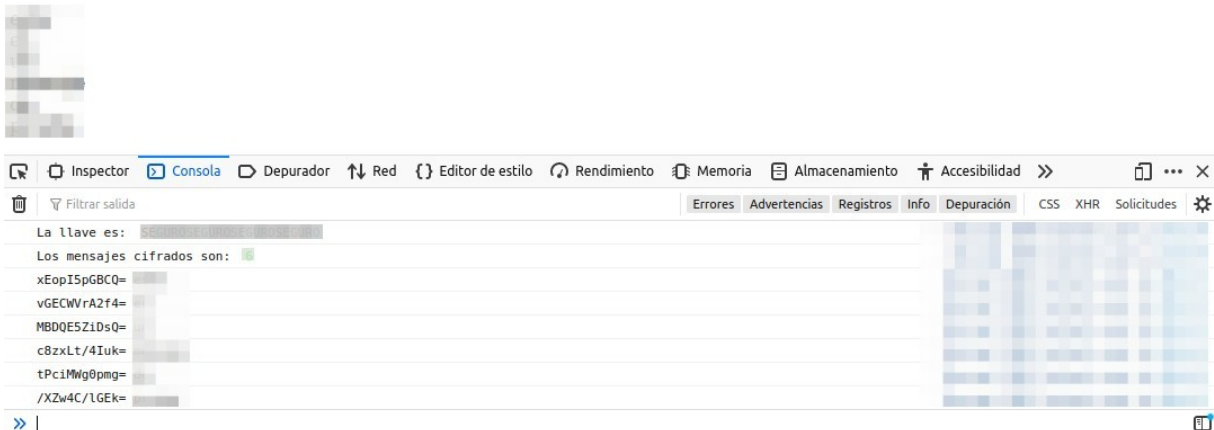
Sólo un ojo entrenado como el suyo logrará descifrar cuál es el algoritmo de cifrado utilizado y cuál es la contraseña utilizada para lograr obtener la información que está oculta.

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.



## 2. Desarrollo (Parte 1)

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

Para comenzar ingresamos al sitio <https://cripto.tiiny.site/> y nos dirigimos al modo de desarrollador dentro en la pestaña Inspector para observar si existe algún factor que llame la atención o resalte frente al resto, y que nos pueda ayudar en la detección del cifrado.

Si nos fijamos dentro del cuerpo de la página, podemos observar unas divisiones curiosas, que poseen un patrón como se puede ver a continuación en la Figura 1:

```
▶ <div id="xEopI5pGBCQ=" class="M1">... </div>
▶ <div id="vGECWVrA2f4=" class="M2">... </div>
▶ <div id="MBDQE5ZiDsQ=" class="M3">... </div>
▶ <div id="c8zxLt/4Iuk=" class="M4">... </div>
▶ <div id="tPciMWg0pmg=" class="M5">... </div>
▶ <div id="/XZw4C/lGEk=" class="M6">... </div>
```

Figura 1: Cifrado

Podemos identificar claramente que cada división posee un ID que sigue el formato clásico de un cifrado en Base64, sobre todo por la presencia de su padding terminando en =. Además de esto, en la Figura 2 podemos ver que el cuerpo de la página existe principalmente de un párrafo encapsulado por **p** como se ve a continuación:

```
▼ <body> [scroll]
  ▼ <p>
    Sin el conocimiento de informaci3n secreta, el
    criptoan3lisis se dedica al estudio de sistemas
    criptogr3ficos con el fin de encontrar debilidades
    en los sistemas y romper su seguridad. El
    criptoan3lisis es un componente importante del
    proceso de creaci3n de criptosistemas s3lidos.
    Gracias al criptoan3lisis, podemos comprender los
    criptosistemas y mejorarlos identificando los puntos
    d3biles. Un criptoanalista puede ayudarnos a
    trabajar en el algoritmo para crear un c3digo
    secreto m3s seguro y protegido. Resultado del
    criptoan3lisis es la protecci3n de la informaci3n
    cr3tica 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 criptogr3fica.
    Sin el conocimiento de informaci3n secreta, el
    criptoan3lisis se dedica al estudio de sistemas
```

html > body

Figura 2: Contenido párrafo

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

Si observamos detalladamente, este consiste de varias oraciones cortas separadas por puntos seguidos, en donde cada una parte con mayúscula y luego de un tiempo, se comienza a repetir desde el inicio. Lo que nos permite inferir que en este sitio de práctica que hay una clave por detrás. Tras prueba y error, se dedujo que al unir cada mayúscula dentro del párrafo se produce una palabra que se va repitiendo, que corresponde a la llave o contraseña del mensaje con un tamaño de 24 bytes.

Por lo tanto como tenemos la presencia de una llave y además se tienen mensajes cifrados podemos inferir que el mensaje del informante se encuentra en cifrado simétrico.

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

Para que los scripts que se utilizarán durante el informe funcionen exclusivamente en el sitio en análisis, necesitamos explicitarlo dentro de los parámetros iniciales en el preámbulo del script de TamperMonkey, como se muestra a continuación.

```
1 // ==UserScript==
2 // @name          Lab 4 cripto
3 // @namespace      http://tampermonkey.net/
4 // @version        1.0
5 // @description    Identifica mensajes cifrados y los descifra
6 // @author         aleudp
7 // @match          https://cripto.tiiny.site/
8 // @grant          none
9 // ==/UserScript==
```

Código 1: Parámetros iniciales

En el Código 1 podemos ver los parámetros iniciales del script encapsulados por **==UserScript==**, en donde definimos el título, el autor y su descripción. Pero lo que interesa en este momento es la restricción **@match** que nos indica que sitio va a ser afectado por este script.

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

Con el análisis previamente realizado, sabemos que hay que juntar en orden todas las mayúsculas dentro del párrafo principal del sitio para obtener la llave escondida. Para hacer esto creamos nuestra primera función mostrada a continuación en el Código 2:

```
1 function encontrarLlave() {  
2   var textElement = document.querySelector('p');  
3  
4   if (!textElement) {  
5     return 'No se encontró el elemento de texto';  
6   }  
7  
8   var text = textElement.textContent;  
9   var mayuscula = text.match(/[A-ZÁÉÍÓÚÑ]/g);  
10  var llave = mayuscula ? mayuscula.join('') : 'No hay mayusculas';  
11  
12  return llave;  
13 }  
14 var llave = encontrarLlave();  
15 console.log("La llave es:", llave);
```

Código 2: Función para encontrar la Llave

Donde podemos ver que consiste en buscar el o los párrafos encapsulados en **p** dentro del sitio, y va concatenando todas las mayúsculas dentro del texto, ignorando las que posean tildes o caracteres especial agregados por errores de encoding. Finalmente la función retorna el valor de la llave.

## 2.4. Muestra la llave por consola

Al ejecutar la función del Código 2 mediante TamperMonkey, podemos ver directamente en la consola del sitio en el modo desarrollador el resultado, esto se debe a que guardamos el resultado de la función en una variable llamada llave y luego la imprimimos mediante `console.log()`.

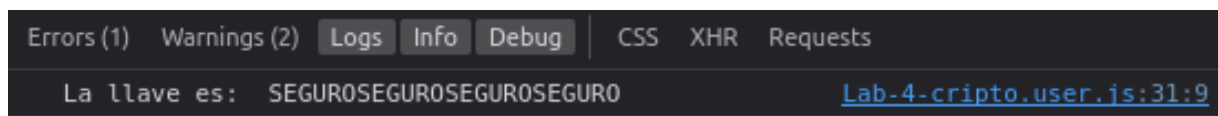


Figura 3: Llave

En la Figura 3 se puede observar la respuesta en consola que posee la llave generada, que consiste en la palabra "SEGUROSEGUROSEGUROSEGURO" de 24 bytes de largo, ya que cada caracter ASCII es de 8 bits.

## 3. Desarrollo (Parte 2)

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

Para reconocer la cantidad de mensajes automáticamente creamos una nueva función. Como vimos anteriormente en la Figura 1, se supuso que cada mensaje cifrado se encontraba dentro de una división, y que cada una de éstas posee una clase que comienza con M seguida de un número.

```
1 function encontrarMensajesCifrados() {  
2     var mensajes = document.querySelectorAll('[class^="M"]');  
3     console.log("Los mensajes cifrados son: ", mensajes.length);  
4  
5     mensajes.forEach(function(contenidoMensajes) {  
6         var id = contenidoMensajes.id;  
7         var contenidoCifrado = contenidoMensajes.textContent.trim();  
8         console.log(id);  
9     });  
10 }
```

Código 3: Función para identificar mensajes cifrados

En el Código 3 podemos ver nuestra segunda función, que busca y lee todas las clases que contienen la letra M y añade su contenido (id) a un arreglo, seguido de esto, le indicamos que imprima el largo de este arreglo que corresponde a la cantidad de mensajes cifrados.

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

En la Figura 4 se puede ver el resultado en consola de la cantidad de mensajes al llamar la función `encontrarMensajesCifrados()` creada en el Código 3.



Figura 4: Cantidad de mensajes

Que nos indica que hay 6 mensajes cifrados, que lo podemos comprobar con el contenido de la Figura 4 que iba desde M1 a M6.

La función creada, además de agregar todos los ID de las clases a un arreglo e imprimir su tamaño, va iterando sobre todo el arreglo y va imprimiendo su contenido, que corresponden a los mensajes cifrados. A continuación en la Figura 5 se puede ver el resultado en consola de la cantidad de mensajes.

xEopI5pGBCQ=	<a href="#">Lab-4-cripto.user.js:40:17</a>
vGECWVrA2f4=	<a href="#">Lab-4-cripto.user.js:40:17</a>
MBDQE5ZiDsQ=	<a href="#">Lab-4-cripto.user.js:40:17</a>
c8zxLt/4Iuk=	<a href="#">Lab-4-cripto.user.js:40:17</a>
tPciMWg0pmg=	<a href="#">Lab-4-cripto.user.js:40:17</a>
/XZw4C/lGEk=	<a href="#">Lab-4-cripto.user.js:40:17</a>

Figura 5: Mensajes cifrados

En donde se ven los 6 mensajes codificado en Base64.

## 4. Desarrollo (Parte 3)

### 4.1. Importa la librería cryptoJS

Para poder descifrar el contenido oculto en el sitio, utilizaremos la librería CryptoJS, para hacer esto nos dirigimos al sitio <https://cdnjs.com/libraries/crypto-js> y obtenemos el link correspondiente a la librería para poder utilizarla de manera online en TamperMonkey.

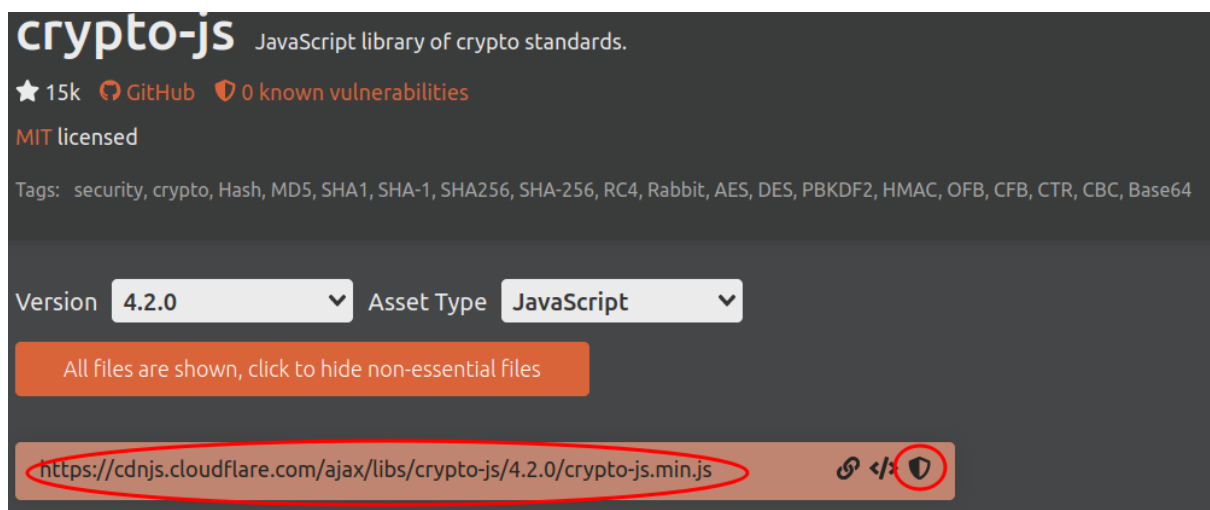


Figura 6: Link CryptoJS

En la Figura 6 podemos ver como obtenemos el link de la librería que se encuentra encerrado en el círculo izquierdo de la imagen.



Para poder importarlo a TamperMonkey, lo agregamos a los parámetros iniciales con el comando de @require, como se ve en el Código 4 a continuación.

```
1 // ==UserScript==
2 // @name      Lab 4 cripto
3 // @namespace  http://tampermonkey.net/
4 // @version   1.0
5 // @description Identifica mensajes cifrados y los descifra
6 // @author    aleudp
7 // @match     https://cripto.tiiny.site/
8 // @grant     none
9 // @require   https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/
10 //          crypto-js.min.js
11 // ==/UserScript==
```

Código 4: Importación CryptoJS

## 4.2. Utiliza SRI en la librería CryptoJS

Se utiliza SRI (subresource integrity) como complemento de la librería CryptoJS, que consiste en una herramienta que nos permite verificar que no hubo manipulación del texto por un tercero mediante el uso de un Hash. Para utilizarlo le hacemos click al link encerrado en el círculo derecho de la Figura 6. Y éste a su vez se importa a TamperMonkey concatenándolo al link de la librería CryptoJS con un símbolo hash entre medio, como se ve en el Código 5 a continuación:

```
1 // ==UserScript==
2 // @name      Lab 4 cripto
3 // @namespace  http://tampermonkey.net/
4 // @version   1.0
5 // @description Identifica mensajes cifrados y los descifra
6 // @author    aleudp
7 // @match     https://cripto.tiiny.site/
8 // @grant     none
9 // @require   https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/
10 //          crypto-js.min.js#sha512-a+SUDuwnzXDvz4XrIcXHuCf089/
11 //          iJAoN4lmrXJg18XnduKK6YlDHNRAlv4yd1N400KI80tFidF+rqTFKGPoWFQ==
12 // ==/UserScript==
```

Código 5: Utilización SRI

Que dentro de todo lo que dice, se puede identificar que utiliza el hash sha512.

## 4.3. Repercusiones de SRI inválido

Como se mencionó en el punto anterior, utilizamos SRI para identificar si hubo una manipulación de terceros del texto, si utilizamos un SRI inválido, no se podrá verificar correctamente si es que hubo o no esta manipulación de los datos, ya que los Hash no van a

coincidir y en consecuencia, el script no se cargará y se mostrarán errores en la consola del navegador exactamente como se ve en la Figura 7 a continuación.

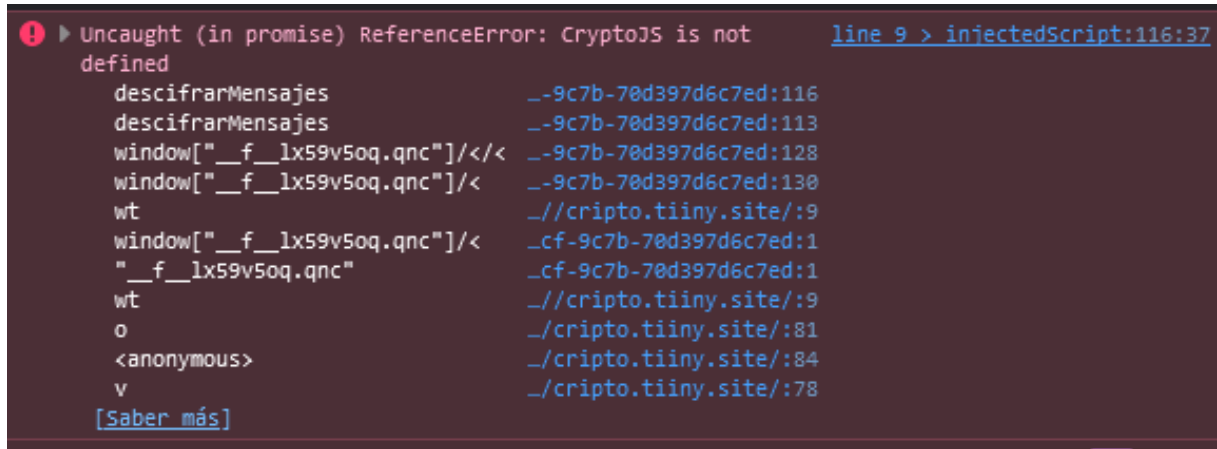


Figura 7: Error SRI inválido

En donde se indica que la librería CryptoJS no está definida, ya que no corroboró el Hash correspondiente.

#### 4.4. Logra descifrar uno de los mensajes

Para descifrar el primer mensaje que se encuentra con cifrado simétrico, podemos crear la siguiente función mostrada en el Código 6 a continuación:

```

1  function descifrarPrimerMensaje() {
2    var mensajes = document.querySelectorAll('[class^="M"]');
3    console.log("Los mensajes cifrados son: ", mensajes.length);
4
5    if (mensajes.length > 0) {
6      var contenidoMensaje = mensajes[0];
7      var id = contenidoMensaje.id;
8
9      var mensajeDescifrado = CryptoJS.TripleDES.decrypt({
10         ciphertext: CryptoJS.enc.Base64.parse(id)
11       }, CryptoJS.enc.Utf8.parse(llave), {
12         mode: CryptoJS.mode.ECB,
13         padding: CryptoJS.pad.Pkcs7
14       }).toString(CryptoJS.enc.Utf8);
15
16       console.log(id + ' ' + mensajeDescifrado);
17       contenidoMensaje.innerHTML = mensajeDescifrado;
18     } else {
19       console.log("No se encontraron mensajes para descifrar.");
20     }
21 }

```

## Código 6: Definición de la estructura del grafo

Esta función consiste en obtener el primer ID dentro del arreglo de todas las clases que poseen M en su nombre, de manera similar al proceso realizado en los códigos anteriores. Teniendo el primer mensaje, lo desciframos utilizando 3DES, que se seleccionó iterando y teniendo en cuenta que la llave poseía un tamaño de 24 bytes.

Ejecutando esta función, nos da como resultado sólo el contenido del primer mensaje impreso en consola como se ve en la Figura

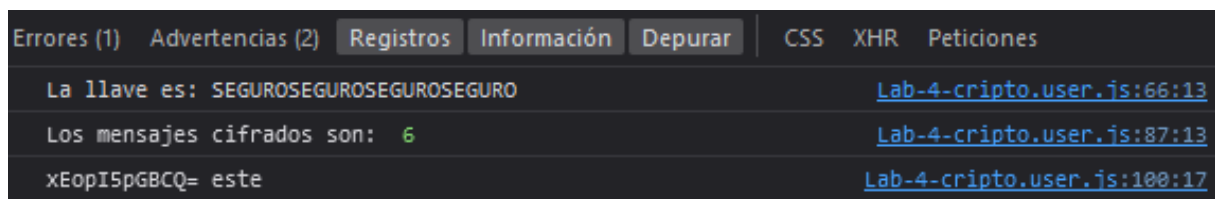


Figura 8: Primer mensaje descifrado

## 4.5. Imprime todos los mensajes por consola

Para imprimir todos los mensajes, se realiza una simple modificación al Código anterior, para que el programa vaya iterando sobre todos los IDs dentro del arreglo, y los vaya descifrando uno a uno utilizando la llave obtenida. Además esta nueva función se combina con la del Código 1, para que además de descifrar los mensajes, sea capaz de imprimir la cantidad total de mensajes y su contenido cifrado, como se ve a continuación en el Código 7.

```

1  encontrarMensajesCifrados();
2
3  function descifrarMensajes() {
4  var mensajes = document.querySelectorAll('[class^="M"]');
5
6  mensajes.forEach(function(contenidoMensajes) {
7  var id = contenidoMensajes.id;
8
9  var mensajeDescifrado = CryptoJS.TripleDES.decrypt({
10 ciphertext: CryptoJS.enc.Base64.parse(id)
11 }, CryptoJS.enc.Utf8.parse(llave), {
12 mode: CryptoJS.mode.ECB,
13 padding: CryptoJS.pad.Pkcs7
14 }).toString(CryptoJS.enc.Utf8);
15
16 console.log(id + ' ' + mensajeDescifrado);
17 contenidoMensajes.innerHTML = mensajeDescifrado;
18 });
19 }

```

Código 7: Definición de la estructura del grafo

Al ejecutar el Código 7, obtenemos los mensajes descifrados como se ve en la Figura 9

xEopI5pGBCQ= este	<a href="#">Lab-4-cripto.user.js:59:9</a>
vGECWVrA2f4= es	<a href="#">Lab-4-cripto.user.js:59:9</a>
MBDQE5ZiDsQ= un	<a href="#">Lab-4-cripto.user.js:59:9</a>
c8zxLt/4Iuk= mensaje	<a href="#">Lab-4-cripto.user.js:59:9</a>
tPciMWg0pmg= de	<a href="#">Lab-4-cripto.user.js:59:9</a>
/XZw4C/lGEk= prueba	<a href="#">Lab-4-cripto.user.js:59:9</a>

Figura 9: Mensajes descifrados

Donde observamos que el mensaje oculto era **este es un mensaje de prueba**.

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

Para lograr mostrar el mensaje descifrado en texto plano en el sitio web, lo único que hay que realizar es agregarle la última línea del Código 7 que corresponde a:

- `contenidoMensajes.innerHTML = mensajeDescifrado;`

Que consite en que a medida que se van imprimiendo en la consola, también se imprimen dentro de la página en formato HTML. A continuación se puede ver en la Figura 10 el contenido de los mensajes impresos directamente en el sitio web.

de informaci3n 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 creaci3n de criptosistemas s3lidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos d3biles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un c3digo secreto m3s seguro y protegido. Resultado del criptoanálisis es la protecci3n de la informaci3n cr3tica 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

Figura 10: Mensaje descifrado en texto plano

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

Necesitamos un nuevo sitio web, que posea un párrafo diferente, es decir una llave distinta y una cantidad de mensajes cifrados diferentes al caso visto durante la experiencia.

Comenzamos redactando el mensaje, que como sabemos cada mayúscula contendrá nuestra llave que debe ser de 24 caracteres. Escogemos una llave que consista en la palabra **HOLA** repetida 6 veces, y escribimos el siguiente mensaje que se repetirá esa misma cantidad de veces en el cuerpo del sitio.

Habia una vez un estudiante que tenia un poder de Organización increbile, el no podia entender como era tan bueno para planear sus trabajos con tanta antelación. Lamentablemente todo esto se acabo cuando tomó el ramo de criptografía y seguridad en redes, ya que los laboratorios gastaban mucho tiempo y la materia era difcil de enteder. Aun así, el estudiante tuvo que seguir trabajando, extrañando todo el tiempo libre que tenia anteriormente.

Escribimos este texto en formato HTML dentro de un párrafo como se ve en la Figura 11

```
<head><script defer data-domain="cripto.tiiny.site" src="https://analytics.tiiny.site/js/plausible.js"></script></head><p>
Habia una vez un estudiante que tenia un poder de Organización increbile, el no podia entender como era tan bueno para plan
</p>
```

Figura 11: Texto en html

Y luego lo hosteamos dentro del mismo sitio del ejemplo del laboratorio en <https://tiiny.host>, generando la página con dominio <https://lab4cripto.tiiny.site/> y la inspeccionamos en modo desarrollador como se ven la Figura 12.

#### 4.7 El script logra funcionar con otro texto y otra cantidad de DESARROLLO (PARTE 3)

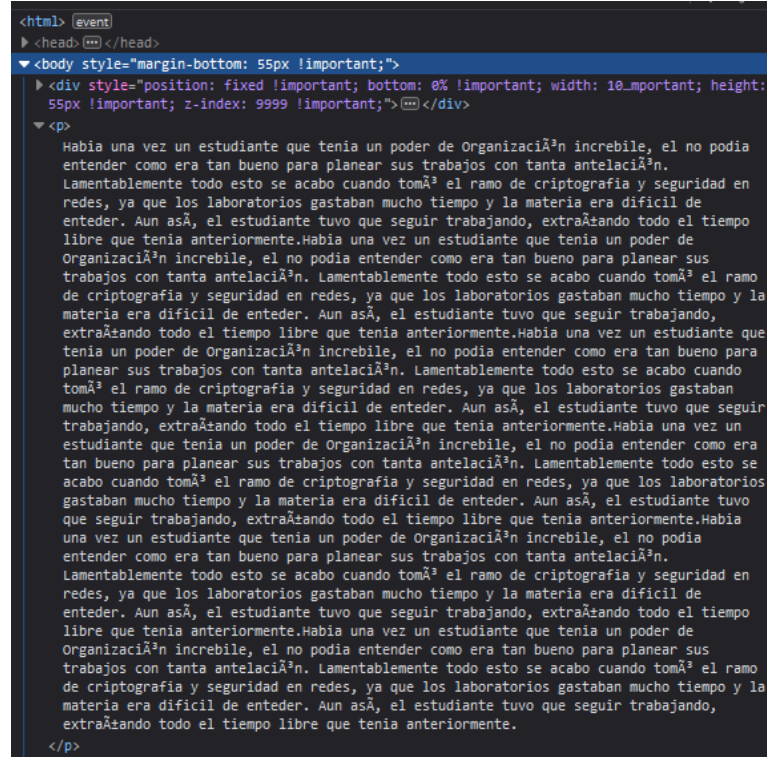


Figura 12: Nuevo párrafo dentro del sitio

Luego necesitamos crear el mensaje y cifrarlo mediante 3DES con codificación en Base64. Para esto creamos una nueva función dentro de nuestro script sin olvidar agregar la nueva url en los parámetros de éste. A continuación se ve el Código 8 conteniendo la nueva función.

```
1 function cifrarMensaje(parrafo, llave) {
2   if (document.querySelector('.M1')) {
3     return;
4   }
5   var mensaje = parrafo.split(" ");
6   var mensajeCifrado = [];
7   mensaje.forEach(function(contenidoMensaje) {
8     contenidoMensaje = CryptoJS.enc.Utf8.parse(contenidoMensaje);
9
10    var cifrado = CryptoJS.TripleDES.encrypt(contenidoMensaje,
11      CryptoJS.enc.Utf8.parse(llave), {
12        mode: CryptoJS.mode.ECB,
13        padding: CryptoJS.pad.Pkcs7
14      });
15    mensajeCifrado.push(cifrado.toString());
16  });
17  var i = 1;
18  mensajeCifrado.forEach(function(palabra) {
19    var div = document.createElement('DIV');
20    div.classList.add('M${i}');
    div.id = palabra;
```

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

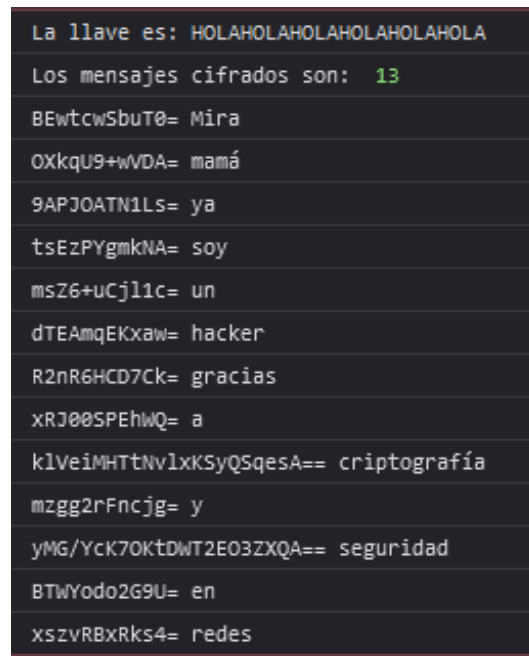
DESARROLLO (PARTE 3)

```
21         document.body.append(div);
22         i++;
23     });
24     return mensajeCifrado.toString();
25 }
```

Código 8: Definición de la estructura del grafo

Esta función recibe como parámetros el texto que se quiere cifrar y la llave. Primero verifica si existe un cifrado previo, preguntando si es que existe la clase llamada M1, que correspondería al primer mensaje cifrado en el ejemplo trabajado del laboratorio. En caso de que no exista, separa el mensaje recibido en palabras, las cifra utilizando la librería CryptoJS usando 3DES, las codifica en base64 y finalmente crea las divisiones dentro del cuerpo de la página con el formato que vimos anteriormente, en clases que consisten en M + número y su ID corresponde a cada palabra cifrada en orden de sintaxis.

Al ejecutar nuestro script, podemos ver tanto en consola como en el texto plano dentro del nuevo sitio web el mensaje descifrado. Primero vemos todos los parámetros y el resultado en consola en la Figura 13.



```
La llave es: HOLAHOLAHOLAHOLAHOLA
Los mensajes cifrados son: 13
BEwtcwSbut0= Mira
OXkqU9+wVDA= mamá
9APJOATN1Ls= ya
tsEzPYgmKNA= soy
msZ6+uCjl1c= un
dTEAmqEKxaw= hacker
R2nR6HCD7Ck= gracias
xRJ00SPehwQ= a
klVeimHTtnvlxKSyQSqesA== criptografía
mzgg2rFncjg= y
yMG/Yck7OKtDWT2E03ZXQA== seguridad
BTWYodo2G9U= en
xsZvRBxRks4= redes
```

Figura 13: Nuevo resultado consola

En donde vemos que la llave es **HOLAHOLAHOLAHOLAHOLA**, que son 13 mensajes cifrados en codificación Base64 y el mensaje es **Mira mamá ya soy un hacker gracias a criptografía y seguridad en redes**. Finalmente en la Figura 14 podemos ver como se muestra en texto plano dentro del cuerpo del sitio web.

Habia una vez un estudiante que tenia un poder de Organizaci3n increbile, el n tiempo y la materia era difcil de enter. Aun as, el estudiante tuvo que seguir antelaci3n. Lamentablemente todo esto se acabo cuando tom3 el ramo de cript anteriormente.Habia una vez un estudiante que tenia un poder de Organizaci3n gastaban mucho tiempo y la materia era difcil de enter. Aun as, el estudiant trabajos con tanta antelaci3n. Lamentablemente todo esto se acabo cuando tom anteriormente.Habia una vez un estudiante que tenia un poder de Organizaci3n gastaban mucho tiempo y la materia era difcil de enter. Aun as, el estudiant trabajos con tanta antelaci3n. Lamentablemente todo esto se acabo cuando tom anteriormente.

Mira  
mamá  
ya  
soy  
un  
hacker  
gracias  
a  
criptografía  
y  
seguridad  
en  
redes

Figura 14: Nuevo texto plano descifrado

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

Los links relevantes para este laboratorio son los siguientes:

1. Script en greasyfork <https://greasyfork.org/en/scripts/497316-lab-4-cripto>
2. Script, cuerpo página HTML en Github <https://github.com/aleudp/Criptografia/tree/ec5874fe90b5fad137f5832b896c27bd65df4df6/Laboratorio%204>
3. Sitio web práctica laboratorio <https://cripto.tiiny.site/>
4. Sitio web de prueba <https://lab4cripto.tiiny.site/>

## Conclusiones y comentarios

Durante este laboratorio pudimos descifrar un mensaje oculto con cifrado simétrico, para esto tuvimos que obtener una llave que estaba escondida mediante la concatenación de todas las mayúsculas dentro del párrafo de un sitio web, mientras que el mensaje cifrado se encontraba dentro de los IDs de clases, siendo una clase para cada palabra dentro del mensaje. Dado el tamaño de la llave se pudo inferir que el método de cifrado correspondía a triple DES, ya que poseía un multiplo de 8 bytes (24).

Teniendo identificado estos elementos, se aprendió y aplicó el uso de javascript mediante tampermonkey, para modificar el sitio web automáticamente. Para el descifrado se tuvo que investigar y ver cómo utilizar la librería de Javascript llamada CryptoJS que nos permitió trabajar con los cifrados 3DES y codificar en Base64.

A medida que se avanzaba en el informe se fue iterando en el script para que abarque las distintas funciones de identificar, a descifrar hasta imprimir el texto plano dentro de la



misma página y finalmente se probó el script creando un nuevo sitio web que cumpliera con el mismo tipo de formato, de modo que este script funciona de manera universal, para todos los sitios que puedan llegar a tener este tipo de cifrado, es decir, no es un script hardcodeado para un caso en particular.

En conclusión, durante el laboratorio pudimos ver como funciona tanto un cifrado y su descifrado simétrico en un ejemplo simple de resolver.

## Issues

Problemas presentados durante el desarrollo del laboratorio.

- **Validación de SRI Inválido:** Se tuvo problemas el manejo de la librería CryptoJS, sobretodo en aplicar SRI, ya que no era claro como se debía importar este complemento. Esto nos impedía la correcta carga y ejecución del script. Solución: Asegurar que el hash SRI utilizado corresponda exactamente a la librería importada y concatenarlo mediante un signo hash.
- **Falta de conocimiento de html:** Inicialmente se tuvieron problemas para identificar los mensajes ocultos y sobre todo la llave. No se recordaba como se podía acceder a los distintos componentes dentro del cuerpo de la página, y en general había una falta de práctica de desarrollo web. Se solucionó practicando, a prueba y error y en fin recordando que cada elemento de los sitios en HTML tiene una subdivisión con nombre y se pueden acceder fácilmente una vez identificadas.
- **Cifrado simétrico:** Fue mas costoso realizar el cifrado que el descifrado, ya que se debía cifrar de una manera específica con 3DES y Base64, esto en fin se solucionó repasando la teoría del funcionamiento del triple cifrado y como se va utilizando la llave en su proceso. Este problema surge simplemente es diferente la aplicación práctica que la teórica.
- **Falta de práctica Javascript:** Similarmente al segundo punto, hubieron problemas de implementación en Javascript simplemente por la falta de práctica, ya que para mi caso particular sólo lo he utilizado en un ramo, varios semestres atrás. Tomó un poco de tiempo recordar el lenguaje para poder aplicarlo correctamente. Se solucionó mediante un conjunto de referencias en linea, inteligencia artificial y práctica al ir escribiendo el código.