

Informe Laboratorio 4

Sección 3

Ignacio Guajardo
e-mail: ignacio.guajardo_m@mail.udp.cl

Noviembre de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (Parte 1)	3
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
3. Desarrollo (Parte 2)	6
3.1. reconoce automáticamente la cantidad de mensajes cifrados	6
3.2. muestra la cantidad de mensajes por consola	6
4. Desarrollo (Parte 3)	6
4.1. Importa la librería cryptoJS	6
4.2. Utiliza SRI en la librería CryptoJS	7
4.3. Logra decifrar uno de los mensajes	7
4.4. Imprime todos los mensajes por consola	8
4.5. Muestra los mensajes en texto plano en el sitio web	8
4.6. El script logra funcionar con otro texto y otra cantidad de mensajes	9
4.7. Indica url al código .js implementado para su validación	10

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.

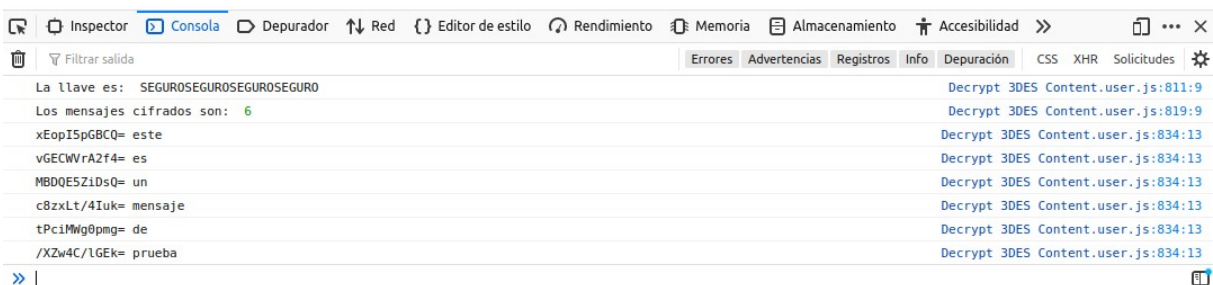
1. Desarrolle un plugin para tampermonkey que permita obtener la llave para el descifrado de los mensajes ocultos en la página web. La llave debe ser impresa por la consola de su navegador al momento de cargar el sitio web. Utilizar la siguiente estructura:
 - La llave es: KEY
2. En el mismo plugin, 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)

2.1. Detecta el cifrado utilizado por el informante

Como se puede ver en el texto de la pagina proporcionada, se tiene un texto en repetición el cuál uno puede asumir que el informante quiere hacer una prueba utilizando el cifrado 3DES, entonces para validarlo se hara lo siguiente. Se inspeccionara los elementos de la pagina y se obtendrá lo siguiente:

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

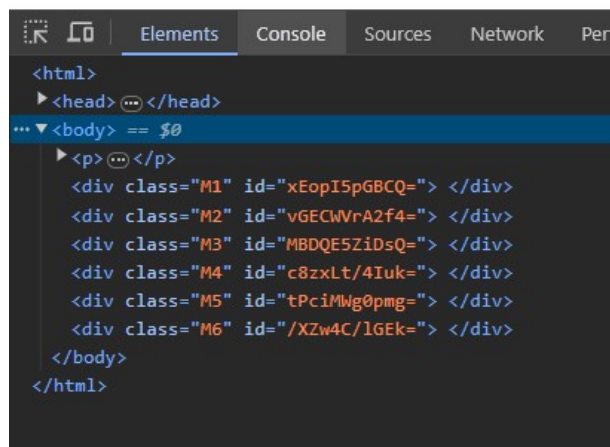


Figura 1: Mensajes Cifrados

Como se evidencia en la imagen adjunta, se presenta una categoría denominada "Parrafo" que engloba el texto previamente mencionado. Además, hay seis divisiones adicionales con las etiquetas "M1", "M2"... "M6", cada una de las cuales contiene un mensaje cifrado en su identificación.

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

Para la realización de este script, se utilizará la extensión de Tampermonkey para poder inyectar código de Javascript en la página otorgada. Para hacer que este script sólo funcione en la página utilizada por el informante, hay que incluir la siguiente línea de código dentro del script:

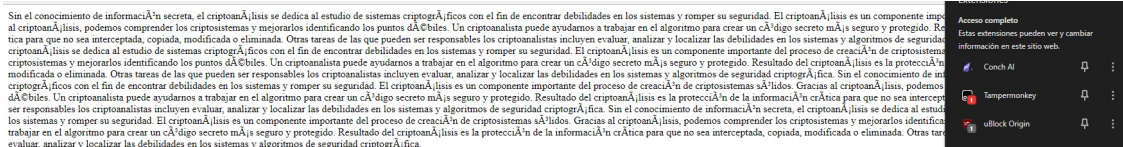
```
// @match      https://cripto.tiiny.site/
```

De forma que, en primera instancia, el código queda de la siguiente forma:

```
// ==UserScript==
// @name        Lab cripto
// @namespace    http://tampermonkey.net/
// @version      0.1
// @description  cripto
// @author       Ignacio
// @match        https://cripto.tiiny.site
// @icon         https://www.google.com/s2/favicons?sz=64&domain=tiiny.site
// @grant        none
// ==/UserScript==
```

Luego, se puede verificar que el script está funcionando correctamente en la página al ver la extensión de Tampermonkey la cuál aparece con un 1 en rojo:

2.3 Define función que obtiene automáticamente el password DESARROLLO (PARTE 1)



2.3. Define función que obtiene automáticamente el password del documento

La contraseña se obtiene mediante la adición de una función al código que examina el texto de los elementos en la página y agrega todas las letras mayúsculas encontradas a una cadena que se utilizará como contraseña. Este código se presenta en la siguiente figura.

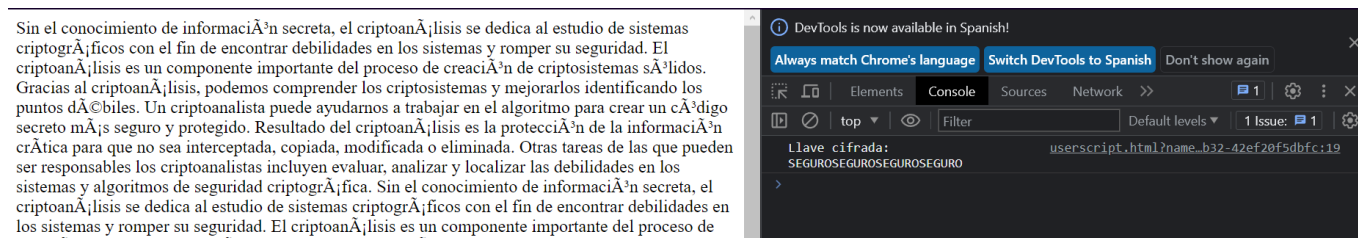
```
function d() {
    const t = document.body.textContent;
    const e = t.match(/[A-Z]/g);

    if (e) {
        const n = e.join('');
        console.log('Llave cifrada:', n);
        return n;
    } else {
        console.log('No se encontraron letras');
        return "Vacío";
    }
}

document.addEventListener('DOMContentLoaded', d);
```

2.4. Muestra la llave por consola

Tras ejecutarse la funcion anterior se muestra la llave por consola, como se observa en la siguiente imagen, mostrando que la llave es "SEGUROSEGUROSEGUROSEGURO".



3. Desarrollo (Parte 2)

3.1. reconoce automáticamente la cantidad de mensajes cifrados

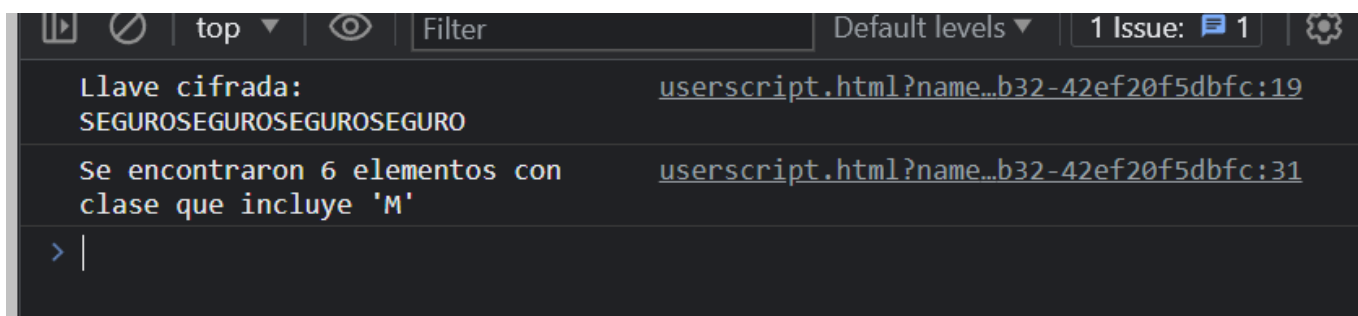
Según la imagen del ítem anterior (2.1), cada mensaje cifrado está ubicado dentro del cuerpo del documento, en divisiones con las clases "M1", "M2"... "M6", siguiendo este patrón, donde el ID de cada una corresponde a un mensaje codificado en base 64. Por ende, para obtener automáticamente la cantidad de mensajes cifrados, se requiere una función capaz de extraer los valores de los IDs de todas las divisiones cuya clase sigue el formato "Mx", donde x es un número entero. Para lograrlo, se emplea la siguiente función:

```
function e(t) {
  const n = document.querySelectorAll('[class*="M"]');
  const o = n.length;

  console.log(`Se encontraron ${o} elementos con clase que incluye 'M'`);
}
```

3.2. muestra la cantidad de mensajes por consola

Para mostrar la cantidad de mensajes por consola, simplemente se debe imprimir el largo del array obtenido con la función del ítem anterior, obteniendo el siguiente resultado:



4. Desarrollo (Parte 3)

4.1. Importa la librería cryptoJS

Para importar la librería de CryptoJS en el script de Tampermonkey, hay que agregarlo como "requerido" en los ajustes del script, haciendo referencia al url de algún CDN que hostee este script. En este caso se agregó lo siguiente:


```
// @require https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/crypto-js.min.js#s
```

4.2. Utiliza SRI en la librería CryptoJS

Para emplear Subresource Integrity (SRI) con la librería CryptoJS, simplemente se copia el valor de SRI proporcionado por la página de cdnjs y se agrega en los encabezados del script de Tampermonkey de la siguiente manera. En este caso, el SRI Hash provisto por CDNJS fue “sha512-a+SUDuwN zXDvz4XrIcXHuCf08 9/iJAoN4lmrXJg 18XnduKK6YlDHNral v4yd1N40OKI80tFidF+rqTFKGPoWFQ==”. Quedando de esta manera:

```
// ==UserScript==
// @name      Lab cripto
// @namespace  http://tampermonkey.net/
// @version   0.1
// @description  cripto
// @author    Ignacio
// @match     https://cripto.tiiny.site
// @icon      https://www.google.com/s2/favicons?sz=64&domain=tiiny.site
// @require   https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/
cripto-js.min.js#sha512-a+SUDuwNzXDvz4XrIcXH
uCf089/iJAoN4lmrXJg18XnduK
K6YlDHNralv4yd1N40OKI80tFidF+rqTFKG
PoWFQ==
// @grant     none
// ==/UserScript==
```

4.3. Logra decifrar uno de los mensajes

Para lograr descifrar cada uno de los mensajes, es necesario crear una función que reciba estos mensajes y la llave correspondiente para descifrarlos. Así, se empleará la siguiente función:

```
function descriptarTexto(clave) {
  const elementosConClaseM = document.querySelectorAll('[class="M"]');
  const numeroElementos = elementosConClaseM.length;

  console.log(`Los mensajes cifrados son :${numeroElementos} `);

  const configuracion = {
    mode: CryptoJS.mode.ECB
  };

  elementosConClaseM.forEach(elemento => {
    const idCifrado = elemento.id;
    const claveCifrado = CryptoJS.enc.Utf8.parse(clave);

    const idDescriptado = CryptoJS.TripleDES.decrypt(idCifrado, claveCifrado, configuracion);

    console.log(`ID: ${elemento.id}, Texto Descriptado: ${idDescriptado.toString(CryptoJS.enc.Utf8)}`);
  });
}
```

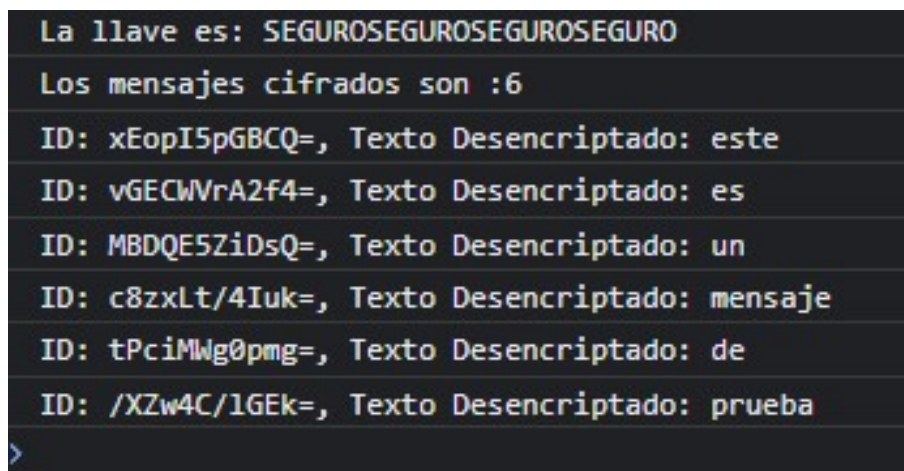
El script utiliza el modo ECB junto al algoritmo de cifrado 3DES para procesar los datos. ECB divide el mensaje en bloques iguales y cifra cada bloque de forma independiente, lo que, aunque sencillo, puede mostrar patrones en el texto cifrado si hay repeticiones en el mensaje original, lo que supone un riesgo para la seguridad.

El algoritmo 3DES, usado para cifrar y descifrar, es una versión mejorada del estándar DES, aplicando un proceso de cifrado triple para mayor seguridad. Aunque anteriormente popular, hoy en día se prefiere el algoritmo AES por su eficiencia y robustez.

El código toma la clave, la formatea en UTF-8 y la emplea para descifrar mensajes recibidos. Utiliza un bucle para procesar los mensajes, convirtiéndolos a Base64 y luego aplicando la función de descifrado TripleDES de CryptoJS. El texto resultante se convierte a texto plano utilizando la función `toString()` con formato UTF-8. Se implementa en la página y se muestra el resultado en la consola para verificar su funcionamiento.

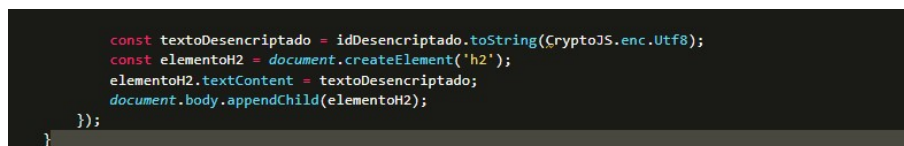
4.4. Imprime todos los mensajes por consola

Como se puede ver en la imagen siguiente, la consola muestra el mensaje, que en este caso sería 'este es un mensaje de prueba'.



```
La llave es: SEGUROSEGUROSEGUROSEGURO
Los mensajes cifrados son :6
ID: xEopI5pGBCQ=, Texto Descifrado: este
ID: vGECWVrA2f4=, Texto Descifrado: es
ID: MBDQE5ZiDsQ=, Texto Descifrado: un
ID: c8zxLt/4Iuk=, Texto Descifrado: mensaje
ID: tPciMWg0pmg=, Texto Descifrado: de
ID: /XZw4C/lGEk=, Texto Descifrado: prueba
>
```

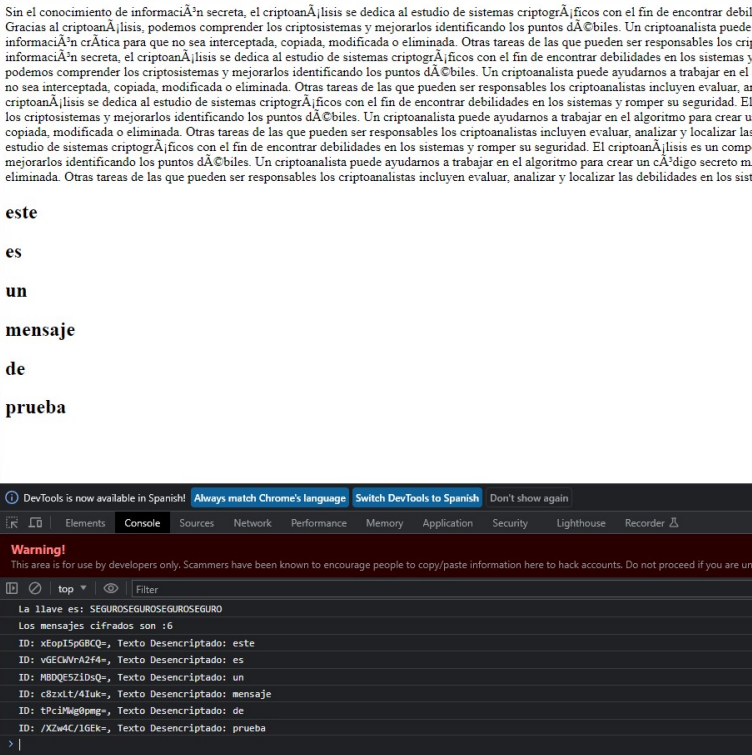
4.5. Muestra los mensajes en texto plano en el sitio web



```
const textoDescifrado = idDescifrado.toString(CryptoJS.enc.Utf8);
const elementoH2 = document.createElement('h2');
elementoH2.textContent = textoDescifrado;
document.body.appendChild(elementoH2);
});
}
```

El código crea un nuevo elemento HTML de nivel de encabezado `h2`. Luego, asigna el texto descifrado a este elemento `h2`, preparándolo para ser mostrado en la página web. Finalmente, añade este elemento `h2`, que contiene el texto descifrado, al cuerpo de la página para que sea visible a los visitantes como se muestra a continuación.

DESARROLLO (PARTE 3)



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

Se creó una nueva página basada en el HTML original. Para esta página, se generó un nuevo texto de manera que la nueva clave fuera 'LABLABLABLABLABLABLAB'. Además, se modificó la cantidad de mensajes a 3. Estos mensajes fueron encriptados con un segundo script de Tampermonkey, utilizando los mismos métodos de encriptación de la librería CryptoJS. Para desencriptarlos, se utilizó el mismo código, pero se ajustó el comando @match para coincidir con la nueva URL.

