Informe Laboratorio 4

Sección 1

Felipe Gutiérrez Lazo
e-mail: felipe.gutierrez_l@mail.udp.cl

Noviembre de 2023

Índice

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:

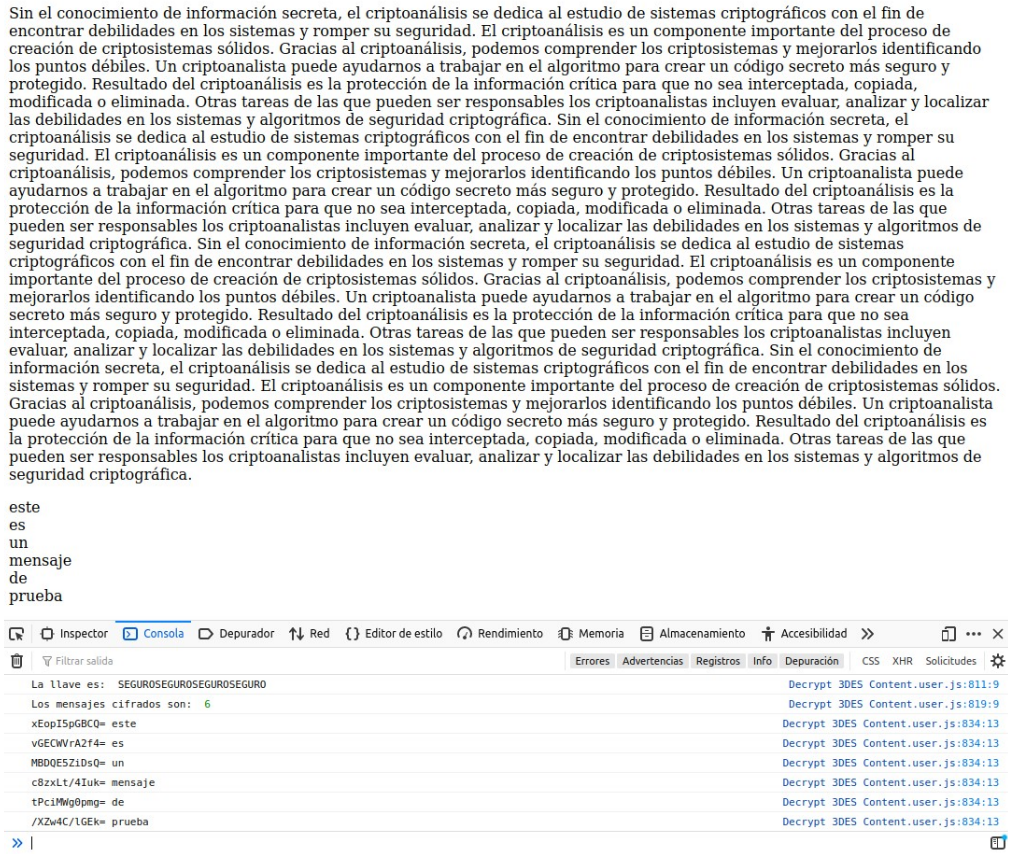
   - La llave es: KEY

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:



2. Desarrollo (Parte 1)

Para poder desarrollar la actividad, es necesario instalar la extensión ***Tampermonkey*** dentro del navegador. Esta es una herramienta que permite modificar los scripts de usuario. Estos scripts de usuario son pequeños programas que son ejecutados en el navegador, y permiten generar modificaciones en la forma en que se despliega una página web, agregando nuevas funciones o automatizando tareas repetitivas. En este caso, Esta extensión será utilizada para obtener mensajes ocultos dentro de un texto plano en un archivo HTML.

2.1. Detecta el cifrado utilizado por el informante

Este paso es el más complejo dentro de la actividad. Para poder detectar el cifrado utilizado por el informante, se debe tener conocimiento sobre los distintos tipos de cifrados existentes. Existen dos grupos de separación, el cifrado simétrico y el cifrado asimétrico. En este caso, el cifrado es simétrico. Uno de los cifrados simétricos más conocidos es el algoritmo DES y sus variantes, 2DES y 3DES. Cada uno de estos necesita una llave para el procedimiento de cifrado y descifrado, y cada llave para cada variante es de una longitud distinta.

Es decir, para DES, la longitud de la llave corresponde a 7 bytes, mientras que 2DES tiene una llave de 114 bytes. Finalmente, 3DES contiene una llave de 24 bytes.

Este dato es crucial para la obtención del algoritmo de cifrado, ya que la llave de este caso corresponde a un string de 24 bytes, que coincide con el largo de la llave de 3DES.

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

Para que el script creado con Tampermonkey solo se gatille en el sitio específico, se describe dentro de los encabezados el metadato ***@match***, que le indica a Tampermonkey la URL o dirección de la página/páginas donde se ejecuta el script. En este caso, se le indica a este metadato que la URL donde debe ser ejecutado corresponde a *https://cripto.tiiny.site/*. De esta manera, el script que está siendo manipulado sólo se ejecuta en la dirección indicada anteriormente.

2.3. Define función que obtiene automáticamente el password del documento

Al momento de entrar en la página web *https://cripto.tiiny.site/*, se encuentra un texto plano común y corriente. Este contiene información que el informante desea transmitir pero de forma camuflada. Luego de analizar el texto, y tratar de descubrir la manera en que el informante hace el envío de la información, se percibe un patrón dentro del texto: las mayúsculas de cada oración forman la palabra **SEGURO** 4 veces. Ya con esta información, es que en conjunto con ChatGPT se crea un programa que permite concatenar cada mayúscula del texto para completar la llave secreta. A continuación, se adjunta una imagen del código que permite realizar esto.

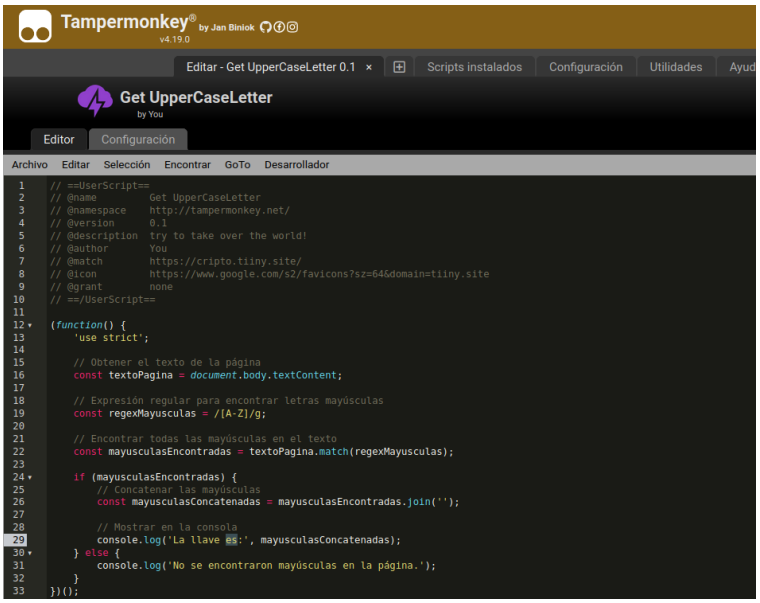


Figura 1: Código para concatenar mayúsculas del texto.

2.4. Muestra la llave por consola

Ya con el código implementado dentro de Tampermonkey, se guarda y se recarga la página que está siendo modificada. Una vez sucede esto, se observa la llave secreta impresa por consola, lo que permite obtener el primer paso para descifrar la información que la persona informante desea enviar. A continuación se adjunta la imagen representativa de esto.

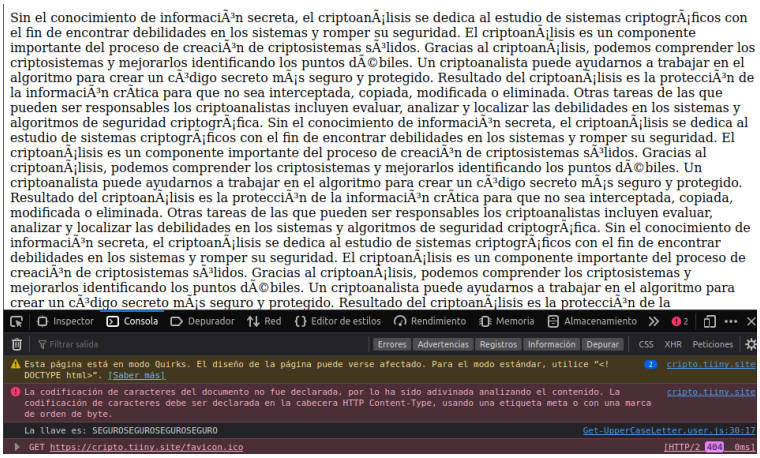


Figura 2: Impresión mediante consola de llave secreta obtenida.

3. Desarrollo (Parte 2)

3.1. reconoce automáticamente la cantidad de mensajes cifrados

Para lograr esto primero se debe identificar la procedencia de los mensajes cifrados. Luego de buscar por la página y el texto plano sin resultado, se comienza a buscar en la estructura de la página. A través de la inspección de la página, se observan 6 contenedores **div** que contienen como identificador un mensaje cifrado. A continuación, se adjunta una imagen ilustrativo de esto.

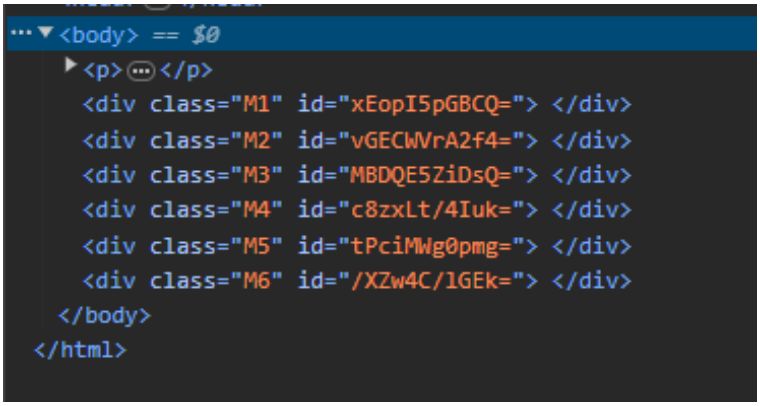


Figura 3: Mensajes cifrados desde inspección a página web.

Además, como se observa en la figura 3, cada div tiene una clase asignada, **Mx**, siendo x cada uno de los mensajes correspondientes. Así, se sabe que la cantidad de mensajes cifrados son 6.

3.2. muestra la cantidad de mensajes por consola

Con el paso anterior ya completado, se procede a generar un código que permita ilustrar la cantidad de mensajes cifrados presentes en la página mediante la consola. Esto se añade al mismo script trabajado para la sección 2. Se adjunta la imagen demostrativa a continuación.

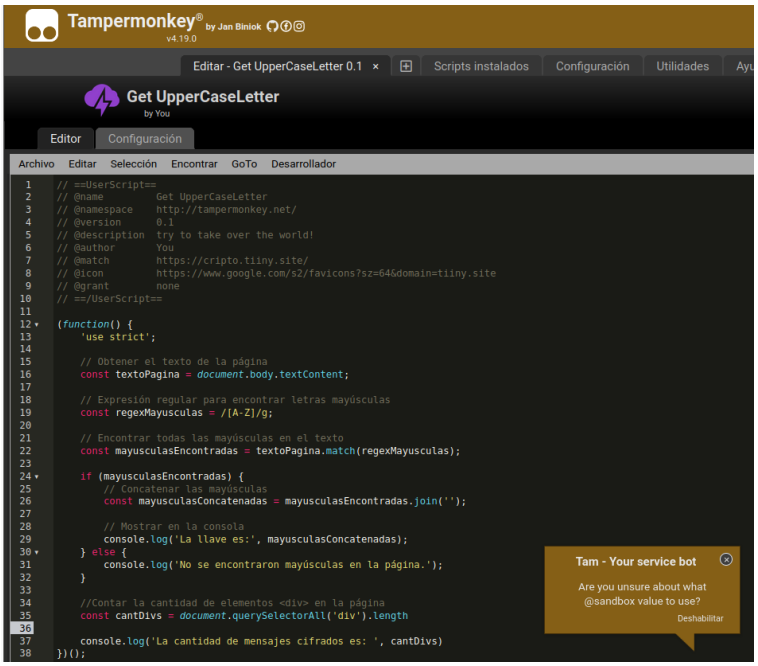


Figura 4: Código para contar cantidad de mensajes cifrados.

Al implementar este código en el script de Tampermonkey, y refrescar la página sobre la que se está trabajando, se puede observar que, en la consola, aparece un mensaje indicando la cantidad de mensajes cifrados, equivalentes a la cantidad de elementos div dentro de la página del informante. Las instrucciones dentro del código que permiten esto corresponden a la línea 35 de la figura 5, que cuenta la cantidad de elementos que son de interés. La imagen de la salida se encuentra a continuación.

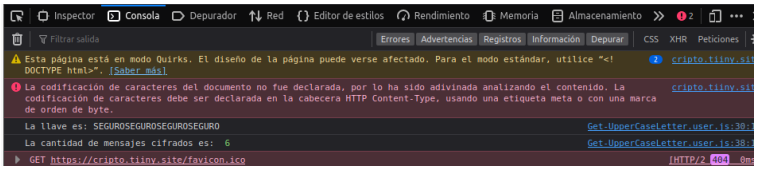


Figura 5: Cantidad de mensajes cifrados desplegados por consola.

4. Desarrollo (Parte 3)

4.1. Importa la librería cryptoJS

Para este apartado, es necesario agregar un header dentro del script trabajado en Tampermonkey. Este header corresponde a **@require**, el cuál indica un script o archivo que debe ser cargado antes de ejecutar el script actual. Es decir, precarga archivos que pueden contener información para la correcta ejecución del código dentro del script actual. Así, a este encabezado se le entrega la URL de la librería **CryptoJS**. Esta se obtiene de **cdnjs**, página que

almacena y contiene librerías de JavaScript. A continuación, se adjunta la imagen ilustrativa de este paso.

Figura 6: Importación de librería CryptoJS.

Ya con el link adjunto al header correcto, se importa la librería que permite realizar los trabajos correspondientes a la parte 3.

4.2. Utiliza SRI en la librería CryptoJS

Para este paso, primero se define que es el SRI y la importancia que tiene.

- **SRI**: Subresource Integrity. Corresponde a la configuración de la integridad del recurso para una librería. Es una técnica de seguridad que verifica la integridad de los archivos mediante una función hash, generando una huella digital única del contenido del archivo.
- **Importancia**: Permite proteger contra ataques como inyección de script y envenenamiento de la caché. Garantiza además la integridad del contenido, verificando que el archivo cargado es el mismo que se distribuye originalmente.

El SRI se obtiene, al igual que la librería CryptoJS, de **CDNjs**. Para poder implementarlo en el código, se debe concatenar luego de la URL que indica la dirección de la librería. El proceso es el siguiente:

- Agregar header necesario. Es el mismo header para el paso 4.1
- En el header, se inserta la URL de la librería a utilizar.
- Luego de agregar la URL, se concatena con esta un #. Se debe añadir esto para poder especificar la huella digital única de una librería.
- Posterior al "#", se agrega el SRI.

En este caso, para la versión utilizada de CryptoJS que corresponde a la versión 4.2.0, el SRI equivale a la expresión **sha512-a+SUD...** . En la figura 6 se observa la implementación de lo recién explicado.

4.3. Logra decifrar uno de los mensajes

Ahora, se crea un programa que permita descifrar los mensajes que se encontraron encriptados dentro de la página. Para lograr esto, se debe tener en posesión una llave para poder realizar el descifrado. Esta llave es la que se extrae desde el contenido de la página, específicamente de las mayúsculas del texto. Todo este procedimiento está detallado en la subsección 2.3. Posteriormente, se debe realizar la decodificación del mensaje cifrado. Al analizar los mensajes cifrados, se observa que se encuentran en una codificación Base 64. Finalmente, se realiza el descifrado de los mensajes. El código implementado es el siguiente.

```
78      // Función para descifrar un mensaje cifrado con 3DES en modo ECB
79      function descifrarMensajeCifrado(llave, mensajeCifradoBase64) {
80          // Convierte la llave de texto a un objeto WordArray
81          const llaveWordArray = CryptoJS.enc.Utf8.parse(llave);
82
83          // Decodificar el mensaje cifrado de Base64 a un objeto WordArray
84          const mensajeCifradoWordArray = CryptoJS.enc.Base64.parse(mensajeCifradoBase64);
85
86          // Realizar el descifrado
87          const mensajeDescifradoWordArray = CryptoJS.TripleDES.decrypt(
88              {
89                  ciphertext: mensajeCifradoWordArray,
90              },
91              llaveWordArray,
92              {
93                  mode: CryptoJS.mode.ECB,
94                  padding: CryptoJS.pad.Pkcs7,
95              }
96          );
97
98          return mensajeDescifradoWordArray;
99      }
100 })();
```

Figura 7: Código utilizado para el descifrado de los mensajes.

Los mensajes están cifrados con 3DES y con un modo de operación ECB. Mediante investigaciones, y pruebas, se logra extraer información sobre el cifrado 3DES. Es sabido que los cifrados DES utilizan llaves para generar el cifrado y el proceso inverso. Es así como, por ejemplo, DES utiliza una llave de longitud equivalente a 7 bytes, mientras que 2DES utiliza una llave con longitud igual a 16 bytes. 3DES utiliza una llave con 24 bytes de longitud. La llave obtenida es **SEGURO**, palabra compuesta por 6 letras. Esta palabra en la llave se repite 4 veces, y calculando y teniendo en cuenta que cada letra equivale a un byte, se obtiene que la llave es de 24 bytes. Mediante este análisis se obtiene que el algoritmo de encriptación es 3DES. El modo de operación se obtiene mediante prueba y error; primeramente se utiliza CBC y los mensajes descifrados no son mostrados ni en consola ni en la página.

El que los mensajes no logren ser descifrados con el modo de operación CBC tiene su lógica, ya que para cifrar con este modo se requiere un vector de inicialización válido que sirva para cifrar y descifrar. En este caso, el valor del vector de inicialización está siendo calculado al azar, por lo que difícilmente coincida con el vector posiblemente utilizado.

Además, el modo de operación EBC da resultados positivos a las pruebas realizadas. Al implementar este modo de operación sobre los mensajes para descifrar, se observan los mensajes en claro en la consola. A continuación, se adjunta la imagen ilustrativa.

Figura 8: Mensaje descifrado.

4.4. Imprime todos los mensajes por consola

Al ver que con el código implementado anteriormente funciona el descifrado de los mensajes, se procede a utilizar con la totalidad de mensajes cifrados encontrados. El resultado de esto se observa en la siguiente imagen.

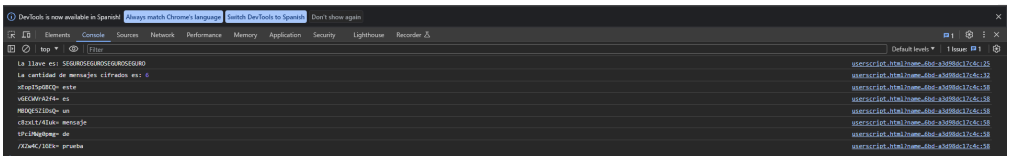
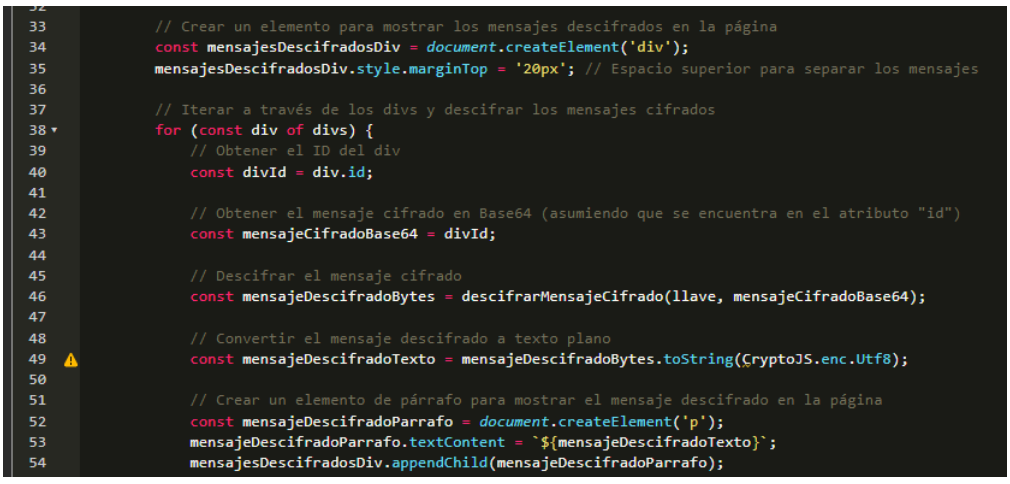


Figura 9: Mensajes descifrados.

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

Ya con los mensajes descifrados, es que se agrega otro extracto de código para que esta vez los mensajes descifrados sean visibles desde el contenido de la página web. El código para esto es el siguiente.



```
// Crear un elemento para mostrar los mensajes descifrados en la página
const mensajesDescifradosDiv = document.createElement('div');
mensajesDescifradosDiv.style.marginTop = '20px'; // Espacio superior para separar los mensajes

// Iterar a través de los divs y descifrar los mensajes cifrados
for (const div of divs) {
    // Obtener el ID del div
    const divId = div.id;

    // Obtener el mensaje cifrado en Base64 (asumiendo que se encuentra en el atributo "id")
    const mensajeCifradoBase64 = divId;

    // Descifrar el mensaje cifrado
    const mensajeDescifradoBytes = descifrarMensajeCifrado(llave, mensajeCifradoBase64);

    // Convertir el mensaje descifrado a texto plano
    const mensajeDescifradoTexto = mensajeDescifradoBytes.toString(CryptoJS.enc.Utf8);

    // Crear un elemento de párrafo para mostrar el mensaje descifrado en la página
    const mensajeDescifradoParrafo = document.createElement('p');
    mensajeDescifradoParrafo.textContent = `${mensajeDescifradoTexto}`;
    mensajesDescifradosDiv.appendChild(mensajeDescifradoParrafo);
```

Figura 10: Mensajes descifrados ilustrados en la página web.

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

Para lograr que funcione con otro texto y otra cantidad de mensajes, se crea una función que agrega al documento divs, con un id específico y una clase específica también. A partir de esto, se hace una inyección de información mediante JavaScript, específicamente de 3 mensajes cifrados más, que corresponden a la frase ƨ resulta perfectamente". El código que permite esto se adjunta a continuación.

```
108     function agregarDiv(id, className) {
109         // Crear un nuevo div
110         const nuevoDiv = document.createElement('div');
111 
112         // Asignar el ID y la clase al div
113         nuevoDiv.id = id;
114         nuevoDiv.className = className;
115 
116         // Agregar el div al final de la página
117         document.body.appendChild(nuevoDiv);
118 
119     }
120 })();
```

Figura 11: Función para inyectar divs al documento.

Los divs que se agregan son los siguientes:

```
26         agregarDiv('b5XKFxSah5Y=', 'M7');
27         agregarDiv('ipd8XWom8BM=', 'M8');
28         agregarDiv('gmylSIHIc8yTe/Jci7K24w==', 'M9');
29 
```

Figura 12: Instrucción para agregar divs a la página

Finalmente, se despliega la información en la consola como en la página. A continuación se adjunta la imagen ilustrativa.

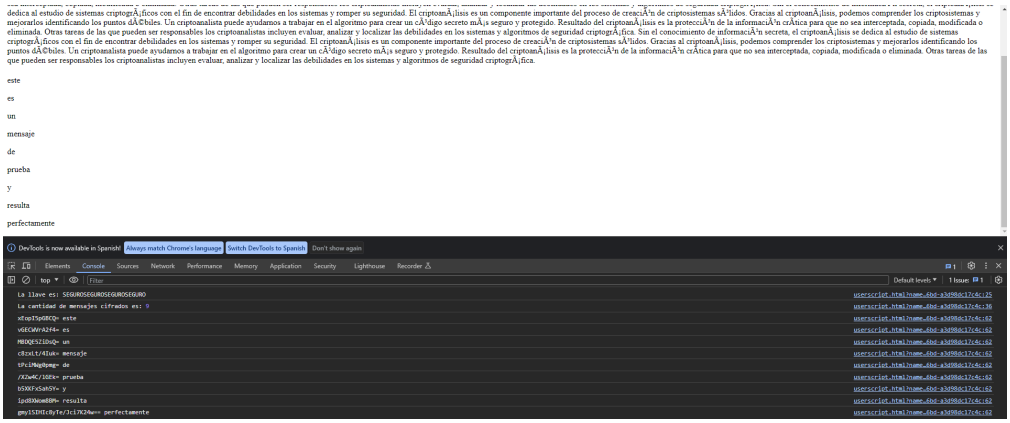


Figura 13: Mensajes agregados desplegados por consola y en el documento.

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

El link para poder acceder al script realizado con Tampermonkey para la experiencia de laboratorio se encuentra en la siguiente URL `https://github.com/felipx28/Lab4_Criptograf-a/blob/main/codigo.js`.

Conclusiones y comentarios

En esta experiencia, se adquiere un conocimiento práctico sobre el cifrado de tipo simétrico con sus distintos algoritmos y modos de operación. De igual manera, se aprende el uso de una nueva herramienta llamada **Tampermonkey** que permite modificar las páginas web mediante un script. Además, se hace uso de la librería **CryptoJS**, que permite utilizar funciones criptográficas con este lenguaje de programación.

Como comentarios, se logra visualizar la poca seguridad que tiene el modo de operación ECB al ser conocida la llave, ya que con ese dato es muy simple descifrar la información. Con modos de operación como CBC o CTR el proceso es más complejo, ya que existen datos que difícilmente son conocidos.