



INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal I.

**Autentificación mediante Chaffing and  
Winnowing en el protocolo HTTP**

2018-B003.

---

Integrantes:

Blancas Pérez Bryan Israel  
Carrillo Fernández Gerardo  
Morales González Diego Arturo  
Paredes Hernández Pedro Antonio

Directores:

Moreno Cervantes Axel Ernesto  
Díaz Santiago Sandra

# Índice.

<b>I Trabajo Terminal I</b>	<b>8</b>
<b>1. Introducción</b>	<b>9</b>
1. Objetivos . . . . .	10
1.1. Objetivo general . . . . .	10
1.2. Objetivos particulares . . . . .	10
2. Justificación . . . . .	11
3. Estado del arte . . . . .	12
4. Metodología . . . . .	13
<b>2. Marco teórico</b>	<b>16</b>
1. Introducción . . . . .	16
2. Métodos de autentificación en internet . . . . .	16
2.1. Cookies . . . . .	19
3. Introducción a la Criptografía . . . . .	21
3.1. Criptología . . . . .	21
3.2. Criptografía . . . . .	21
3.3. Objetivos de la criptografía . . . . .	23
3.4. Usos de la criptografía . . . . .	24
3.5. Criptografía simétrica y asimétrica . . . . .	24
3.6. Criptografía a nivel de aplicación . . . . .	27
4. Chaffing and Winnowing . . . . .	29
4.1. Historia . . . . .	29
4.2. ¿Qué es Chaffing and Winnowing? . . . . .	29
4.3. Objetivo de Chaffing and Winnowing . . . . .	31
4.4. ¿Cómo funciona? . . . . .	32
4.5. Propiedades de Chaffing and Winnowing . . . . .	33
4.6. All-or-Nothing and the Package Transform (AONT) . .	34
4.7. Comparando Chaffing and Winnowing contra Cifrado y Esteganografía . . . . .	36

<b>3. Análisis.</b>	<b>39</b>
1. Estudio de Factibilidad. . . . .	39
1.1. Factibilidad Técnica . . . . .	39
1.2. Factibilidad Operativa . . . . .	41
1.3. Factibilidad Económica . . . . .	42
2. Herramientas a usar. . . . .	43
2.1. Software. . . . .	43
2.2. Hardware. . . . .	47
3. Arquitectura del sistema. . . . .	48
4. Diagrama de casos de uso general. . . . .	49
5. Prototipo I. . . . .	50
5.1. Descripción. . . . .	50
5.2. Estudio de requerimientos. . . . .	50
5.3. Reglas del negocio. . . . .	51
6. Prototipo II. . . . .	52
6.1. Descripción. . . . .	52
6.2. Estudio de requerimientos. . . . .	57
6.3. Reglas del negocio. . . . .	58
7. Prototipo III. . . . .	59
7.1. Descripción. . . . .	59
8. Prototipo IV. . . . .	59
8.1. Descripción. . . . .	59
<b>4. Desarrollo.</b>	<b>60</b>
1. Prototipo I. . . . .	60
1.1. Diagrama de casos de uso. . . . .	60
1.2. Descripción de casos de uso. . . . .	61
1.3. Diagrama de flujo (DF). . . . .	66
1.4. Diagrama de flujo de datos (DFD). . . . .	68
1.5. Diagrama de clases. . . . .	69
1.6. Diagramas de secuencia. . . . .	72
1.7. Diagrama de actividades . . . . .	77
1.8. Interfaz de usuario. . . . .	78
1.9. Requisitos de diseño. . . . .	80
2. Prototipo II. . . . .	82
2.1. Diagrama de casos de uso . . . . .	82
2.2. Descripción de casos de uso. . . . .	83
2.3. Diagrama de flujo (DF). . . . .	92
2.4. Diagrama de flujo de datos (DFD). . . . .	94
2.5. Diagrama de clases. . . . .	96
2.6. Diagramas de secuencia. . . . .	99

2.7.	Diagrama de actividades . . . . .	104
2.8.	Interfaz de usuario. . . . .	105
2.9.	Requisitos de diseño. . . . .	107
<b>5.</b>	<b>Pruebas.</b>	<b>109</b>
1.	Prototipo I. . . . .	109
2.	Prototipo II. . . . .	113
<b>6.</b>	<b>Conclusión</b>	<b>117</b>
<b>7.</b>	<b>Trabajo a futuro (TT2)</b>	<b>119</b>

# Índice de figuras.

2.1. Esquema del protocolo de criptografía simétrica. . . . .	25
2.2. Esquema del protocolo de criptografía asimétrica. . . . .	26
2.3. Charles agrega los paquetes inválidos. . . . .	30
2.4. Charles no agrega los paquetes pero multiplexa los flujos. . . .	31
2.5. Secuencia de Chaffing después del proceso de autentificación. .	32
2.6. Formas del proceso de chaff . . . . .	33
2.7. Visión general del proceso Chaffing and Winnowing. . . . .	33
2.8. Proceso de Chaffing and Winnowing junto con AONT. . . . .	36
3.1. Arquitectura General del Sistema . . . . .	48
3.2. Diagrama de casos de uso general del sistema . . . . .	49
3.3. Arreglo del patrón de chaffing . . . . .	55
3.4. Arreglo del patrón de chaffing después de una iteración . . .	55
3.5. Arreglo del patrón de chaffing final . . . . .	55
3.6. Arreglo Chaff final. . . . .	56
4.1. Diagrama de casos de uso del Prototipo I. . . . .	60
4.2. Diagrama de flujo del Prototipo I. . . . .	66
4.3. Diagrama de flujo de datos del Prototipo I. . . . .	68
4.4. Diagrama de clases del Prototipo I. . . . .	69
4.5. Diagrama de secuencia 1 del Prototipo I . . . . .	72
4.6. Diagrama de secuencia 2 del Prototipo I . . . . .	73
4.7. Diagrama de secuencia 3 del Prototipo I . . . . .	74
4.8. Diagrama de secuencia 4 del Prototipo I . . . . .	75
4.9. Diagrama de secuencia 5 del Prototipo I . . . . .	76
4.10. Diagrama de actividades Del Prototipo I. . . . .	77
4.11. Logo de la extensión. . . . .	78
4.12. Pantalla inicial. Servicio activado. . . . .	79
4.13. Pantalla inicial. Servicio desactivado. . . . .	79
4.14. Nueva pestaña. Encabezado de petición. . . . .	80
4.15. Diagrama de casos de uso del Prototipo II. . . . .	82
4.16. Diagrama de flujo del Prototipo 2. . . . .	92

4.17. Diagrama de flujo de datos de Prototipo 2 . . . . .	94
4.18. Diagrama de clases de Prototipo 2 . . . . .	96
4.19. Diagrama de secuencia 1 del Prototipo II . . . . .	99
4.20. Diagrama de secuencia 2 del Prototipo II . . . . .	100
4.21. Diagrama de secuencia 3 del Prototipo II . . . . .	101
4.22. Diagrama de secuencia 4 del Prototipo II . . . . .	102
4.23. Diagrama de secuencia 5 del Prototipo II . . . . .	103
4.24. Diagrama de actividades del Prototipo 2 . . . . .	104
4.25. Pantalla inicial. . . . .	105
4.26. Tab de la extensión. Permite inicio de sesión . . . . .	106
4.27. Mensaje de éxito . . . . .	106
4.28. Mensaje de error . . . . .	106
4.29. Mensaje de error en certificado . . . . .	107
5.1. Servicio desactivado. Acceso a página <i>www.google.com</i> . . . . .	109
5.2. Servicio desactivado. Acceso a página <i>www.formula1.com</i> . . . . .	110
5.3. Servicio activado. Acceso denegado a <i>www.google.com</i> . . . . .	110
5.4. Servicio activado. Muestra de petición <i>www.google.com</i> . . . . .	111
5.5. Servicio activado. Acceso denegado a <i>www.formula1.com</i> . . . . .	112
5.6. Servicio activado. Muestra de petición <i>www.formula1.com</i> . . . . .	112
5.7. Inicio de sesión para la obtención del certificado. . . . .	113
5.8. Aviso de guardado del certificado en Storage. . . . .	114
5.9. Pagina bloqueada del lado del cliente. . . . .	114
5.10. Wireshark analizando en red WI-FI filtrando análisis por <i>http</i> . . . . .	115
5.11. Análisis de petición en <i>Show Packet Bytes...</i> . . . . .	116

# Índice de cuadros.

2.1. Aplicación de los métodos de autentificación . . . . .	17
2.2. Seguridad en los métodos de autentificación . . . . .	18
3.1. Herramientas de Software . . . . .	40
3.2. Equipo de Hardware 1 . . . . .	40
3.3. Equipo de Hardware 2 . . . . .	40
3.4. Equipo de Hardware 3 . . . . .	41
3.5. Horas de trabajo . . . . .	42
4.1. DCU: PI_CU1 . . . . .	61
4.2. DCU: PI_CU2 . . . . .	62
4.3. DCU: PI_CU3 . . . . .	63
4.4. DCU: PI_CU4 . . . . .	64
4.5. DCU: PI_CU5 . . . . .	65
4.6. DCU: PII_CU1 . . . . .	83
4.7. DCU: PII_CU2 . . . . .	85
4.8. DCU: PII_CU3 . . . . .	86
4.9. DCU: PII_CU4 . . . . .	88
4.10. DCU: PII_CU5 . . . . .	90

# Glosario.

**Cookies** Es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del navegador. Sus principales funciones son recordar accesos y conocer información sobre los hábitos de navegación e intentos de spyware. 19

**Flash** Aplicación informática englobada en la categoría de reproductor multimedia. 44

**HTTP** Hypertext Transfer Protocol. 13–15, 50, 61

**ID** identificador. 16

**Identity theft** También conocido como "robo de identidad" se produce cuando una persona adquiere, transfiere, posee o utiliza información personal de una persona física o jurídica de forma no autorizada, con la intención de efectuar o vincularlo con algún fraude u otro delito. 19

**IU** Interfaz de Usuario. 51

**Netcape** Navegador web de la compañía NetScape Communications. 45

**Single Sign-On** Es un procedimiento de autenticación que le permite a un usuario determinado acceder a varios sistemas con una sola instancia de identificación.. 12

# **Parte I**

## **Trabajo Terminal I**

# Capítulo 1

## Introducción

En la actualidad la mayoría de los usuarios de internet necesitan guardar contraseñas para sus distintas cuentas en las diferentes páginas web a las que ingresan, ya que recordarlas es un problema debido a la gran cantidad de servicios que se utilizan. Como consecuencia de que la autentificación por contraseña es la más utilizada en los servicio web hoy en día [1], los distintos servicios web han implementado mecanismos de seguridad tales como contraseñas que contengan un mínimo de caracteres determinados, al menos un carácter especial, entre otros. Esto ha provocado que éstas sean más difíciles de recordar y han orillado a los usuarios a optar por guardarlas en medios físicos o digitales para recordarlas cuando sea necesario. Sin embargo, perder esas claves (principalmente con los medios físicos) presenta un grave problema de seguridad, teniendo como consecuencia: perdida de datos sensibles, robo de identidad, robo de cuentas bancarias, etc.

La gran mayoría de servicios web han implementado la función "recordar contraseña", la cual hace que el usuario no tenga que ingresar sus credenciales<sup>1</sup> cada vez que se quiere acceder al servicio. Esta función por lo general hace uso de cookies que es información almacenada en el explorador web del usuario, esto representa una amenaza de seguridad ya que permite rastrear la información de navegación del usuario, muy útil para sitios fraudulentos o se puede presentar un robo de cookies con el cual los intrusos podrían hacerse pasar como el usuario, entre otros problemas que éstas representan.

El método *Chaffing and Winnowing* proporciona confidencialidad de los mensajes sin la necesidad de usar métodos de cifrado o esteganografía, agregando 'basura' (*Chaffing*) para que el mensaje quede ininteligible ante la vista de terceros no implicados en la comunicación, y sólo el receptor pueda

---

<sup>1</sup>Credenciales se entiende como los datos que un servicio web requiere para poder acceder al él. Comúnmente son 'usuario' y 'contraseña'.

'limpiar' (*Winnowing*) el mensaje obteniendo la información relevante.

Es por ello que en este trabajo terminal se propone realizar un nuevo método de autenticación por medio del método de *Chaffing and Winnowing* con la ayuda de una extensión de Google Chrome, la cual servirá para la inyección del certificado de autenticación en el protocolo HTTP, emitido por un servidor autenticador que proveerá al usuario un certificado, utilizando una autoridad certificadora basándose en los datos del mismo (Usuario y Contraseña). El usuario obtendrá este certificado al iniciar sesión una única vez en la extensión, y una vez teniendo este certificado, solo será necesario iniciar sesión una vez en cada servicio, ya que el certificado se asociará a esta cuenta y posteriormente cuando el usuario quiera acceder, se validará este certificado para dar respuesta al usuario. Así, si un servicio web tiene este tipo de autenticación disponible, lo podrá validar. Esto es, poder hacer el proceso de *Winnowing* con el cual podrá obtener el certificado original y poder validararlo. El propósito principal de este trabajo es que los usuarios puedan realizar un inicio de sesión más cómodo, seguro y sin la necesidad de recordar sus distintas contraseñas constantemente.

## 1. Objetivos.

### 1.1. Objetivo general.

Realizar un nuevo método de autenticación modificando los datos del protocolo HTTP utilizando Chaffing and Winnowing con la ayuda de una extensión del navegador Google Chrome.

### 1.2. Objetivos particulares.

- Investigar e implementar el desarrollo de extensiones en Google Chrome.
- Investigar sobre los mecanismos de autenticación.
- Investigar sobre la técnica de *Chaffing and Winnowing* para adaptar su implementación.
- Inyectar el código (*Chaffing*) en el encabezado HTTP para enviar la petición al servidor.
- Implementación de un Login en la extensión de Google Chrome.

- Investigar sobre Autoridades Certificadoras para implementar un servidor autentificador
- Generación de certificado autentificador del usuario
- Implementación de un servicio web de prueba.
- Desarrollo de un API para nuestro servidor que obtenga el certificado del usuario (*Winnowing*).
- Modificar el código del servidor Apache para simular y comprobar el funcionamiento de la extensión.
- Realizar pruebas de seguridad para comprobar la eficacia de la extensión.

## 2. Justificación.

Los usuarios de Internet, han optado por guardar sus contraseñas en medios físicos o digitales, lo cual presenta un problema grave de seguridad ya que la pérdida o acceso a uno de estos medios, significaría el acceso a todos los servicios guardados en éstos. Ante esto, la gran mayoría de los servicios web, han implementado una solución al problema de recordar las credenciales y lo tedioso de tener que ponerlas cada vez que se desea acceder, la cual es recordar tu usuario y contraseña, para que automáticamente puedas acceder al servicio. Dicha solución presenta un problema de seguridad, ya que los archivos en donde se guarda la información se pueden copiar y con ello replicarlos en cualquier otra computadora.

Tenemos también algunas opciones que los navegadores y servicios nos ofrecen como el de “recordar contraseña”, sin embargo, estos métodos no son del todo seguros para los usuarios, por ejemplo, la configuración de Google Chrome nos permite observar todas las contraseñas de las diferentes páginas a las que hemos accedido sin ninguna especie de autenticación de quién accede a estas contraseñas, lo cual supone un claro riesgo a la integridad del dueño de estas cuentas.

Es por ello, que este proyecto busca beneficiar al usuario evitando que tenga que ingresar sus credenciales continuamente, y sin que el sistema tenga que guardar las credenciales del servicio web que el usuario use. Por lo tanto, este proyecto brinda una alternativa segura de autenticación en un

servicio web.

### 3. Estado del arte

Como se ha mencionado anteriormente el método de autentificación *usuario y contraseña* debe hacerse siempre que se desee ingresar al servicio, a menos que se le recuerde al usuario la contraseña por medio de cookies. Para esto, IBM ha implementado una autentificación de usuarios mediante el inicio de sesión único, la ultima versión (IBM TRIRIGA 10.6.0 & IBM TRIRIGA Application Platform 3.6.0) fue creada en el año 2018 [29]. Esta autentificación está implementada para obtener acceso a **IBM TRIRIGA**<sup>2</sup>, en esta plataforma es necesario autentificar un usuario como usuario valido del sistema y concederle acceso a las aplicaciones y funciones de la suite<sup>3</sup> de aplicaciones de *IBM TRIRIGA*. El inicio de sesión único (Single Sign-On) proporciona a los usuarios una vía para gestionar el acceso a varias aplicaciones de su entorno.

*TRIRIGA Application Platform* utiliza su propia autentificación nativa de forma predeterminada. Con la autentificación nativa, el usuario especifica su nombre de usuario y su contraseña en una pantalla de inicio de sesión de IBM TRIRIGA, luego entonces, la plataforma autentifica el usuario comparando el nombre de usuario y la contraseña almacenados en la base de datos de IBM TRIRIGA.[28]

El proceso de este tipo de autentificación de TRIRIGA es el siguiente:

- El usuario especifica la URL del servidor web en un navegador o accede a la aplicación mediante un cliente.
- Es posible que se le solicite al usuario que especifique un nombre de usuario o una contraseña, o bien que se inicie la sesión inmediatamente. El inicio de sesión inmediato, sin que el servidor cuestione al navegador o al cliente, no está soportado en algunas configuraciones.
- El servidor web, el servidor de aplicaciones o el plug-in de autenticación verifica la información con el origen de la autenticación.

---

<sup>2</sup>IBM TRIRIGA es un sistema de gestión del espacio de trabajo integrado (IWMS) que incrementa el rendimiento operativo, financiero y medioambiental de sus instalaciones y bienes inmuebles.

<sup>3</sup>El término suite hace referencia a un conjunto de aplicaciones y herramientas de software incluidas en un sólo paquete y que, por lo general, comparten un aspecto similar y se integran entre sí.

- Si el inicio de sesión es satisfactorio, el servidor web añade las credenciales de usuario a la cabecera HTTP y las envía al servidor de aplicaciones.
- El servidor de aplicaciones procesa las credenciales de usuario e inicia la sesión en la aplicación.

## 4. Metodología.

El proceso de desarrollo que seguiremos estará basado en la metodología de prototipos evolutivos, el cual consiste en la implementación parcial del proyecto cumpliendo con los requerimientos que van surgiendo a lo largo del desarrollo, de esta manera es posible ir experimentando con un prototipo parcialmente funcional e identificar posibles mejoras o fallas con el fin de lograr el objetivo final. Esta metodología está compuesta por las siguientes fases:

- Fase de investigación preliminar.
- Especificación de requerimientos y prototipos
- Diseño técnico
- Programación y pruebas
- Operación y mantenimiento

Primero tendremos la fase de “investigación preliminar”, donde se van a definir las metas principales, después en la fase de “especificación de requerimientos y prototipos”, se hace el diseño básico para dar paso a la creación del primer prototipo correspondiente, y después verificar el cumplimiento de los requerimientos y de ser necesario modificarlo hasta que los cumpla. En la tercera fase (diseño técnico) se realiza un diseño detallado y la documentación necesaria para que en la cuarta fase (programación y pruebas) se implemente y se pruebe el prototipo. Finalmente, en la última fase (Operación y mantenimiento) se hace la liberación y el mantenimiento del prototipo final.

Para nuestro proyecto realizaremos 4 prototipos, los cuales son:

1. Creación de extensión de Google Chrome para interceptar la petición HTTP.
  - Investigación preliminar:

- Investigar sobre el desarrollo de extensiones en Google Chrome.
  - Especificación de requerimientos y prototipos:
    - Ejecución de la extensión sobre Google Chrome.
    - Detectar petición HTTP e interceptarla.
    - Subir archivo autentificador a la extensión.
  - Diseño técnico:
    - Documentación del prototipo.
  - Documentación del prototipo.
    - Desarrollo de la extensión.
    - Pruebas de la extensión.
2. Inyección de código autentificador (Chaffing) en el encabezado HTTP.
- Investigación preliminar:
    - Investigación sobre el método Chaffing and Winnowing.
  - Especificación de requerimientos y prototipos:
    - Lectura del archivo autentificador.
    - Análisis del protocolo HTTP.
    - Inyección del código autentificador sobre el protocolo HTTP.
    - Mandar petición a servidor.
  - Diseño técnico:
    - Documentación del prototipo.
  - Programación y pruebas:
    - Desarrollo del complemento de la extensión.
    - Creación del algoritmo de inyección de código.
    - Pruebas de la extensión.
3. Modificación del servidor Apache para recibir el protocolo con la inyección de código.
- Investigación preliminar:
    - Investigación sobre el servidor Apache.
    - Analizar la arquitectura del servidor Apache
    - Investigación sobre la versión conveniente a modificar.
  - Especificación de requerimientos y prototipos:

- Recibir petición HTTP de la extensión.
  - Detectar el tipo de autenticación que se usará.
- Diseño técnico:
    - Documentación del prototipo.
  - Programación y pruebas:
    - Descargar la versión del servidor Apache a usar.
    - Modificación del código del servidor Apache para detectar el tipo de autenticación que se usará.
    - Pruebas de funcionamiento.
4. Realización de la autenticación (Winnowing) en el servidor para realizar el login.
    - Investigación y pruebas
      - Investigar sobre la implementación de autenticador en distintos servidores
    - Especificación de requerimientos y prototipos:
      - Recibir la petición.
      - Descifrar la petición.
      - Dar respuesta al usuario.
    - Diseño técnico:
      - Documentación del prototipo.
    - Programación y pruebas:
      - Creación de algoritmo que obtenga el código autenticador del protocolo HTTP.
      - Verificación del código autenticador.
      - Responder al usuario.
      - Pruebas de funcionamiento.

# Capítulo 2

## Marco teórico

### 1. Introducción.

Dado que este Trabajo terminal relaciona temáticas muy enfocadas a la seguridad y aspectos web, es necesario conocer algunos conceptos e ideas fundamentales tanto para entender el trabajo como para conocer su funcionamiento, por lo tanto será necesario hablar de métodos de autenticación, criptografía y acerca del método *Chaffing and Winnowing* que es la parte modular de todo este trabajo; en este marco teórico se explicará de manera breve y con un enfoque directo el uso que le daremos a estos conceptos en el proyecto.

### 2. Métodos de autenticación en internet.

Con la evolución de la web, distintas páginas ofrecen ciertos servicios a los usuarios y con la finalidad de dar una experiencia óptima y segura, se requiere que una persona o usuario se identifique para el uso de estos servicios, es aquí donde entra en juego el papel de los métodos de autenticación. Para poder asegurar la confidencialidad de la información manejada en todos estos servicios, es necesario restringir el acceso de este, para esto se utiliza la identificación y la autenticación; la identificación es un procedimiento donde el sujeto es reconocido por algún ID, esto es equivalente al saber una parte de información en específico, mientras que la autenticación es el proceso de validación de si el sujeto es la persona quien dice ser al tratar de identificarse.[21]

Para probar una identidad, el sujeto presenta algo llamado "factor de autenticación", principalmente existen 4:

- El sujeto tiene algo (Token, documento, un material específico, etc.).

- El sujeto conoce algo (contraseña, login, respuesta a una pregunta, etc.).
- El sujeto tiene una característica biológica (huella dactilar, ADN, etc.).
- El sujeto se encuentra en un lugar en específico (dirección IP, información de un lugar en específico, etc.).

Hoy en día, la autenticación por contraseña es el método más utilizado, más que otra cosa por su ventaja principal: su simplicidad de utilización, sin embargo, así como tiene una gran ventaja, la autenticación por contraseña también tiene muchos problemas y desventajas de seguridad.

A continuación, mostraremos algunas tablas comparativas que servirán para tener una mejor perspectiva de las ventajas, desventajas, vulnerabilidades de los diferentes métodos de autenticación, entre otras cosas:

	Recordar	Otros dispositivos	Acciones	Facilidad	Tiempo	Errores	Recuperación
Contraseñas	1	3	2	3	3	2	3
Otros recursos	2	3	3	3	3	3	2
Contraseñas gráficas	1	1	2	3	3	2	3
Contraseñas dinámicas	1	3	2	2	3	2	2
Tokens	3	1	1	2	2	3	1
Multivariación	1	1	1	3	2	2	1
Cryptografía	3	1	1	1	1	2	1
Biométricos	3	3	2	3	2	2	1

Cuadro 2.1: Tabla comparativa de la aplicación en los distintos métodos de autenticación

La tabla anterior concentra las siguientes características:

- Recordar: Hace referencia a que tan complicado es que un usuario se acuerde de los datos necesarios para la autenticación.
- Otros dispositivos: El usuario usa una entidad externa para facilitar su autenticación.
- Acciones: Hace referencia a que tantas acciones adicionales se deben de realizar para autenticarse.
- Facilidad: Simplicidad de tecnología.
- Tiempo: Cantidad de recursos temporales que consume el método de autenticación.

- Errores: Posibles errores durante la autentificación.
- Recuperación: Denota la dificultad de recuperar la clave de acceso en caso de pérdida.

En el cuadro No.2 se muestra una tabla comparativa del nivel de seguridad en los distintos métodos de autentificación, donde 1 - baja seguridad, 2 – media seguridad y 3 – alta seguridad.

	Ataque por fuerza bruta	Observación	Hackeo indirecto	Phishing
Contraseñas	1	1	1	1
Otros recursos	2	2	3	3
Contraseñas gráficas	1	1	2	2
Contraseñas dinámicas	2	3	2	2
Tokens	3	3	3	3
Multivariación	1	1	3	3
Criptografía	3	3	3	3
Biométricos	3	3	1	1

Cuadro 2.2: Tabla comparativa de la seguridad en los distintos métodos de autentificación

La tabla se enfoca principalmente en los siguientes problemas de seguridad:

- Ataque por fuerza bruta: Se descifra el método de autentificación con una gran cantidad de intentos, usualmente generados por un programa.
- Observación: Cuando se intenta ver directamente los datos necesarios para la autentificación desde una distancia cercana hasta incluso usando binoculares, cámaras o algún otro dispositivo.
- Hackeo indirecto: El usuario confía sus datos del método de autentificación a terceros quienes pueden ser atacados.
- Phishing: Hace referencia a programas que se hacen pasar por entidades confiables para interceptar los datos que desean.

**Seguridad en internet** En la actualidad, el incremento constante de internet ha impactado directamente en la seguridad de la información que se maneja cotidianamente y por la mayoría de usuarios. Existen infinidad de sitios donde es aplicada la seguridad, ya que sin ésta, se verían afectados

todos los usuarios en sus cuentas, pudiendo verse afectados desde un posible Identity theft (Robo de identidad), hasta la perdida de dinero real dado que la base de algunas de éstas paginas son E-Commerce, estas paginas implican el manejo de tarjetas de crédito, paypal, etc.

Uno de los puntos más críticos de la seguridad en Internet son las herramientas que interactúan de forma directa con los usuarios. Es común escuchar sobre fallas en los sistemas de protección de los servidores más frecuentemente utilizados, por ejemplo Apache, NGINX, IIS, etc. O en los lenguajes de programación en que son escritas las aplicaciones. [9] Sin embargo, la vulnerabilidad más grande dentro de un sistema, son los ataques directos a los usuarios finales durante la autentificación.

Una de las técnicas de autentificación que actualmente se usa es "recordar la sesión" usando las "cookies", en la siguiente sección, nos adentraremos en definir qué son las cookies y exponer sus vulnerabilidades.

## 2.1. Cookies.

Durante la navegación por internet, la información sobre la computadora puede ser colectada y almacenada. Esta puede ser de carácter general sobre el equipo y puede ser también información más específica sobre los hábitos de navegación del usuario, toda esta información guardada se le conoce como Cookies[12].

A continuación se muestran los diferentes tipos de cookies que existen para los navegadores.

- **Cookies propias:** Las cookies se gestionan desde el terminal o dominio de un mismo editor.
- **Cookies de terceros:** Las cookies no son enviadas por el propio editor, sino por otra entidad.
- **Cookies de sesión:** Los datos recabados sólo se recogen mientras el usuario está navegando por la página web.
- **Cookies persistentes:** Los datos continúan almacenados en el terminal y se puede acceder a ellos durante un periodo de tiempo determinados.
- **Cookies técnicas:** Permiten controlar el tráfico y la comunicación de datos.

- **Cookies de personalización:** Dejan a los usuarios acceder según algunas características propias que se recogen (navegador, idioma, etc.).
- **Cookies de análisis:** Recogen datos sobre el comportamiento de los usuarios y permiten elaborar un perfil de usuario.
- **Cookies publicitarias:** Recogen datos sobre la gestión de los espacios publicitarios.

Las cookies persistentes son aquellas que se almacenan en el equipo para que las preferencias personales puedan ser retenidas, ayudan a los sitios web a recordar tu información y ajustes cuando los visitas más adelante. Esto conlleva un acceso más rápido y sencillo ya que, por ejemplo, no se tiene que iniciar sesión de nuevo. Además de la autenticación, otras páginas web tienen más funciones para las cookies permanentes, como: selección de idioma, selección de tema, preferencias de menú, marca-páginas internos de la web, o favoritos. [13] Muchos navegadores pueden ajustar el periodo de tiempo en que las cookies persistentes deben ser almacenadas.

Gracias a las cookies persistentes, las direcciones de correo electrónico aparecen por default cuando se abre el correo electrónico, o en páginas de inicio personalizadas cuando se visita en línea un comercio. Si un atacante obtiene acceso puede recopilar información personal del usuario través de estos archivos y poder robar toda información del usuario. Es fácil acceder a estas cookies y obtener fácilmente la información del usuario, por lo que es necesario que el usuario nunca deje vulnerable esta información o en su debido caso borrar cookies al término de cada sesión. Existen diferentes funcionalidades para las cookies, una de las más importantes es la funcionalidad de seguridad, ya que contiene información importante de los usuarios. A continuación se muestran las diferentes funcionalidades de las cookies.

- **Preferencias:** Sirven para que la página se visualice atendiendo a los gustos del usuario, como por ejemplo idioma, región o tamaño de textos.
- **Seguridad:** Se encargan de autenticar a los usuarios y evitar el uso fraudulento de las credenciales por parte de terceros.
- **Procesos:** Son utilizadas para el correcto funcionamiento de la página en el navegador.
- **Publicitarias/Estadísticas:** Se usan para que la publicidad que se muestre sea personalizada.

- **Estados de la sesión:** Obtienen información del comportamiento del usuario en una página web, como por ejemplo el tiempo que pasa en una página, los "clicks" que realiza o la publicidad que le aparece.

Una vez entendido que es la autentificación y el uso que se le puede dar a las cookies, procedemos a exponer como la criptografía es de gran ayuda en la seguridad en internet y como la usaremos para este nuevo método de autentificación.

### 3. Introducción a la Criptografía.

#### 3.1. Criptología.

Para comprender que es la criptografía, es necesario que comprendamos qué es la "*Criptología*", palabra que proviene del griego «*kryptós*» (oculto) y «*logos*» (estudio). Según la Real Academia Española significa 'estudio de los sistemas, claves y lenguajes ocultos o secretos', sin embargo, brindándole un contexto a esta definición decimos que es el arte y ciencia que se encarga de diseñar sistemas para ocultar mensajes, y buscar la manera de romper dichos sistemas. [2]

La criptología contiene dos ramas, las cuales son: el criptoanálisis y la criptografía. Ésta última es de vital importancia en este trabajo terminal, por lo que a continuación se explicará qué es, su historia, sus objetivos y sus usos que tiene esta rama en la actualidad.

#### 3.2. Criptografía.

"*Criptografía*", palabra proveniente del griego «*kryptós*» (oculto) y «*graphos*» (escribir), es definida por la Real Academia Española como 'el arte de escribir con clave secreta o de un modo enigmático'. Nuevamente, la definición de la RAE nos da un panorama general de lo que trata esta rama de la criptología, sin embargo, en la actualidad tenemos definiciones más extensas y precisas que nos ayudan a entender las funciones de este arte.

A continuación, se presentan dos definiciones de la criptografía, cabe mencionar que estas definiciones están orientadas al uso de la criptografía en la informática y las telecomunicaciones actualmente.

Jorge Ramírez Aguirre nos brinda la siguiente definición en su libro 'Seguridad informática y criptografía'. [3]

*"Rama inicial de las matemáticas y en la actualidad también de la informática y la telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar, y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando una o más claves."*

Finalmente, Menezes, Van Oorschot y Vanstone, no brindan en su libro una definición formal de lo que es la criptografía (traducción). [4]

*"Estudio de técnicas matemáticas relacionadas con los aspectos de la seguridad de la información tales como la confidencialidad, la integridad de datos, la autenticación de entidad y del origen de datos. La criptografía no comprende sólo a los medios para proveer seguridad de información, sino a un conjunto de técnicas."*

A elección del lector elegir aquella definición que le convenza más, nosotros una vez finalizado la exposición de estas definiciones de criptografía, continuaremos con una breve remembranza de la historia de esta disciplina.

En egipto hace 4000 años, tuvo su primera aparición la criptografía, cuando un maestro egipcio escribió la historia de su señor utilizando jeroglíficos poco comunes tratando de imprimir cierta jerarquía. Este primer acercamiento a la criptografía, utilizaba las técnicas de sustitución y transposición de símbolos de una manera similar a la base del concepto general de cifrado. Posteriormente, las antiguas civilizaciones occidentales comienzan a adoptar estas técnicas para mantener determinada información oculta, principalmente con propósitos militares, diplomáticos y religiosos. Mientras la criptografía crecía alrededor del mundo, el criptoanálisis también lo hacía, con el objetivo de hallar el mensaje original a partir de un mensaje cifrado, sin conocer el método utilizado. [5][6]

El la historia conocida después de lo antes mencionado, existe un punto crucial en el desarrollo de la criptografía tal y como la conocemos hoy en día hasta la llegada de las computadoras. Este punto fue la Segunda Guerra Mundial, en donde se construyeron máquinas de cifrado mecánicas y electromecánicas que aceleraban el proceso de cifrado y descifrado. Los alemanes desarrollaron la famosa máquina 'Enigma', que precisamente mediante unos rotores automatizaba el proceso de cifrado y descifrado de mensaje, brindándole al ejército una ventaja considerable en la inversión de tiempo en la comunicación.

### 3.2.1. Criptografía moderna.

En la actualidad, el término '*criptografía moderna*' hace alusión al desarrollo de esta ciencia en las áreas de la teoría de la información y las comunicaciones. Claude Shannon, considerado por muchos como el padre de la criptografía matemática, en su libro "Sistemas secretos" estableció las bases para las implementaciones de los algoritmos actuales a mediados de los años 50s. [6]

En los años 70s, el público general tuvo acceso al trabajo de Claude, además surgió la llegada de las computadoras y la publicación el primer borrador del algoritmo de criptografía simétrica DES, el cual fue el primer algoritmo público, basado en técnicas matemáticas y criptográficas modernas. Todo ello representó los cimientos para un rápido crecimiento en la criptografía, hasta eventualmente llegar a otro hecho sumamente relevante que determinó gran parte de las transacciones que realizamos hoy en día en internet. Dicho hecho fue el artículo de las nuevas direcciones de criptografía hecho por Whitfield Diffie y Martin Hellman, el cual trataba de un nuevo método para distribuir llaves, que eventualmente se llamaría criptografía asimétrica.

## 3.3. Objetivos de la criptografía.

Algunos de los objetivos que se busca con la implementación de la criptografía para la seguridad de la información son los siguientes:

- **Confidencialidad:** este objetivo, también conocido como privacidad de la información, implica mantener en secreto una determinada información, por tanto, sólo aquellas personas que estén autorizadas tendrán acceso a ella.
- **Autenticación:** este objetivo, implica hablar de la corroboración de la identidad de una entidad, por tanto, asegura que la entidad de donde la información es originada pueda ser identificada.
- **Integridad:** este objetivo asegura que determinada información no haya sido alterada por personas no autorizadas o por cualquier otro medio no conocido.
- **No repudio:** este objetivo previene que una entidad niegue un envío de información de un acuerdo preestablecido.

### **3.4. Usos de la criptografía.**

Los usos que tiene la criptografía son muy variadas, dependiendo del ámbito donde se está aplicando. Las siguientes usos, son sólo algunas de los tantos que hay y provienen de distintos objetivos que tiene la criptografía aplicada a la seguridad de la información. [6]

- **Autorización:** permiso concreto, de una parte o entidad, para el acceso o la realización de una tarea específica.
- **Validación:** proveer una autorización para el uso o la manipulación de información o recursos.
- **Control de acceso:** restricción de acceso a la información o recurso.
- **Certificación:** respaldo de información por una entidad externa de confianza.
- **Confirmación:** acuse de recibo a servicios que han sido dados.
- **Anonimato:** ocultamiento de la identidad de una entidad.

### **3.5. Criptografía simétrica y asimétrica.**

Anteriormente en la historia de la criptografía, se hizo una rápida mención del nacimiento de estos dos métodos de cifrado, en esta sección se explicará más profundamente sus funciones con el fin de que el lector conozca un poco más y entienda porque hemos decidido utilizar determinado método para cumplir con los objetivos de este trabajo.

#### **3.5.1. Criptografía simétrica.**

En la criptografía simétrica, tanto el emisor como el receptor comparten una única llave secreta para cifrar y descifrar la información que se deseé transmitir. Esto implica que ambas partes de la comunicación deben tener un acuerdo antes de que se realice la comunicación. La seguridad de este tipo de algoritmos radica en mantener segura la llave secreta, por tanto, si ésta es revelada, cualquiera con acceso a ella puede descifrar el mensaje. Por estas razones, este tipo de criptografía puede ser visto como "criptografía de llave privada". Algunos de los algoritmos más famosos de criptografía simétrica son: DES (Data Encryption Standard), TripleDES, AES (Advanced Encryption Standard) y IDEA (International Data Encryption Algorithm).[17] En el siguiente esquema se muestran los pasos que sigue un protocolo de criptografía simétrica. Definidos 'A' como una entidad que pretende enviar un

mensaje a otra entidad llamada 'B'. Luego entonces, ambas partes acordarán una 'Llave Secreta' con la que 'A' cifrará el mensaje utilizando un algoritmo establecido, mandando el resultado (Texto Cifrado) a 'B' que decifrará el mensaje con la misma llave y algoritmo que 'A'.



Figura 2.1: Esquema del protocolo de criptografía simétrica.

### 3.5.2. Criptografía asimétrica.

La criptografía de clave pública fue inventada en 1976 por los matemáticos Whitfield Diffie y Martin Hellman y es la base de la criptografía moderna. En los algoritmos de criptografía asimétrica, el receptor posee una llave pública y una llave privada para poder descifrar los mensajes. Las claves privadas deben ser conocidas únicamente por su propietario, mientras que la correspondiente clave pública puede ser dada a conocer abiertamente. Si un usuario quiere enviar a otro un mensaje de forma que sólo el receptor pueda entenderlo, lo codificará con la clave pública del receptor y éste utilizará su clave privada, que solo él tiene, para poder leerlo. Dicho esto, podemos llamar llave de cifrado a la llave publica y llave de descifrado a la llave privada, siendo ambas totalmente diferentes una de la otra. Algunos de los algoritmos más famosos de criptografía asimétrica son: RSA y ElGamal. [17]

La criptografía asimétrica está basada en la utilización de números primos muy grandes. Si multiplicamos entre sí dos números primos muy grandes, el resultado obtenido no puede descomponerse eficazmente, es decir, utilizando los métodos aritméticos más avanzados en las computadoras más avanzadas sería necesario utilizar durante miles de millones de años tantas computadoras como átomos existen en el universo. El proceso será más seguro cuanto mayor sea el tamaño de los números primos utilizados.

En el siguiente esquema se muestran los pasos que sigue un protocolo de criptografía simétrica. Definamos 'A' como una entidad que desea enviar

información a otra llamada 'B'. Luego entonces, 'B' enviará a 'A' su llave pública para que 'A' cifre la información utilizando la. Cuando la información (TextoCifrado) haya viajado a través del canal inseguro para que 'B' la reciba, 'B' decifrará el TextoCifrado con su llave privada.



Figura 2.2: Esquema del protocolo de criptografía asimétrica.

### Algoritmo RSA.

*RSA* es el algoritmo de cifrado asimétrico más popular en la actualidad. Creado por Ron Rivest, Adi Shamir y Leonard Adleman y publicado en el año 1977. El algoritmo es considerado seguro, en tanto sean utilizadas llaves de longitud suficientemente seguras (se siguen utilizando llaves de 1024 bits, pero ya se recomienda al menos una longitud de 2048). El algoritmo sirve tanto para cifrar y descifrar, así como también para la generación de firmas digitales. Es, en la actualidad, ampliamente utilizado en protocolos de comercio electrónico. La seguridad *RSA* está basada en la dificultad de realizar el *factoreo* de números grandes. La llave privada y la pública son generadas o calculadas en función de un par de números primos, del orden de los 200 dígitos o más grandes aún. [6]

Al describir ya concretamente al algoritmo, se establece que para la generación del par de llaves (Llave pública y privada) se deberán seleccionar dos números primos grandes aleatorios,  $p$  y  $q$ , y se calculará  $n$  como su producto:

$$n = pq$$

La llave de cifrado,  $e$ , será elegida también de manera aleatoria, tal que  $e$  y  $(p - 1)(q - 1)$  sean primos relativos.

La llave de descifrado  $d$  será obtenida despejando la ecuación:

$$ed = 1 \bmod (p - 1)(q - 1)$$

En otras palabras, o mejor dicho, en otros símbolos:

$$d = e^{-1} \bmod((p - 1)(q - 1))$$

Los números  $e$  y  $n$  componen la llave privada; el número  $d$  corresponde a la llave privada;  $p$  y  $q$  serán descartados pero no revelados.

A la hora de cifrar un mensaje  $m$ , éste deberá ser dividido en bloques más pequeños que  $n$  y cada parte del texto-cifrado,  $c$ , será obtenida mediante:

$$c_i = m_i^e \bmod n$$

Para el descifrado, cada parte o bloque del texto-cifrado se tomará para calcular:

$$m_i = c_i^d \bmod n$$

El patrón con el que se genera el código *Chaffing* no puede viajar a través de la red visible para cualquier entidad que quiera obtener dicha petición, por lo que, debemos cifrarlo con algún algoritmo confiable.

El algoritmo RSA nos proporciona esa seguridad, debido a que es un cifrado asimétrico con el cual podemos cifrar el patrón con la llave pública del servicio. Una vez que la petición cifrada llegue al servicio, este será capaz de descifrar dicho mensaje con su llave privada, para así poder ver el mensaje original.

### 3.6. Criptografía a nivel de aplicación.

La criptografía a nivel de aplicación se entiende como aplicaciones o programas informáticos que hacen uso de diferentes técnicas criptográficas. Al hablar de criptografía en el nivel de aplicación, también podría considerarse a los protocolos criptográficos de alto nivel, los cuales determinan como han de comunicarse ambas partes, que algoritmos usar, define formatos, etc.[6]

#### 3.6.1. SSL/TLS.

Secure Sockets Layer (SSL) provee servicios de seguridad entre la capa TCP y las aplicaciones que hacen uso de esa capa. Actualmente, la sucesora de SSL es TLS (Transport Layer Service), sin embargo, lo acuñado que está el término SSL hace que se use indistintamente para referirse a TLS, aunado a ello, las diferencias entre la última versión de SSL (SSL3.0) y la primera versión de TLS (TLSv1) son menores, por lo que en el desarrollo de este

reporte, utilizaremos el término SSL/TLS.

SSL/TLS provee entonces confidencialidad, lográndola con criptografía asimétrica y controlando la integridad de los datos utilizando un MAC (Message Authentication Code).

El proceso de comunicación del protocolo establece, como primer paso, la negociación de ambas partes de los algoritmos a utilizar. Luego, procede al intercambio de llaves públicas y a la autentificación basada en certificados digitales para, finalmente, cifrar de manera simétrica los datos o información a transferir.[6]

En nuestro proyecto, usaremos SSL/TLS para brindar confidencialidad al canal de comunicación entre la extensión y el servidor autenticador. Esto se explicará más a detalle en el análisis del Prototipo IV.

### 3.6.2. OpenSSL

OpenSSL es un proyecto de código abierto que implementa funciones criptográficas sin limitaciones dentro de una librería y que provee diversas herramientas útiles. OpenSSL actualmente implementa SSL2.0, SSL3.0 y TLSv.[6]

Dentro de las herramientas que tiene se encuentran:

- Crear y manejar llaves privadas, públicas y parámetros.
- Realizar operaciones criptográficas de llave pública.
- Calcular hash de algún mensaje.
- Cifrar y descifrar con algoritmos simétricos.
- Crear certificados X.509 (CSRs y CRLs).

Esta última herramienta, es la que usaremos para este proyecto. Se creará un certificado de cada usuario a partir de sus datos de inicio de sesión para autenticarlo en los servicios web en donde quiera acceder. Esto se explicará más a detalle en el análisis del Prototipo IV.

Ahora que se ha expuesto el uso de la criptografía en este proyecto, procedemos a platicar acerca del método *Chaffing and Winnowing* y cómo se implementará en este trabajo.

## 4. Chaffing and Winnowing.

### 4.1. Historia

Chaffing and Winnowing es una técnica que logra confidencialidad sin usar ningún proceso de cifrado para el envío de datos sobre un canal inseguro. El nombre **Chaffing and Winnowing** el nombre proviene de la agricultura: Después de que el grano ha sido cosechado y trillado, el grano es mezclado con paja fibrosa no comestible. La paja y el grano son separados por el movimiento de las hojas y la paja es descartada. Esta técnica fue creada por Ron Rivest y fue publicada en un artículo en línea el 18 de Marzo de 1998. [20] Aunque parece ser similar a un cifrado tradicional y esteganografía, chaffing and winnowing no puede ser clasificado como uno de ellos. Esta técnica permite el envío de datos evitando la responsabilidad del cifrado de su contenido. Cuando se usa chaffing and winnowing, el emisor transmite el mensaje sin cifrar (texto plano). Aunque el emisor y el receptor comparten una llave, ellos la usan sólo para autenticar. Sin embargo, una tercera parte puede hacer su comunicación confidencial durante el envío simultáneo de mensajes especialmente mensajes diseñados a través del mismo canal.

### 4.2. ¿Qué es Chaffing and Winnowing?

Chaffing and Winnowing es un nuevo esquema establecido por Rivest en 1998. Este esquema ofrece confidencialidad para el contenido de un mensaje sin involucrarse con cifrado ni esteganografía. [17]

El proceso **Chaffing** no hace uso de un cifrado por lo que no tiene una "clave de cifrado". Este proceso consiste en agregar paquetes inválidos (Información innecesaria) al mensaje a enviar, haciendo que el mensaje viaje seguro a la vista de todos los posibles "atacantes".

El proceso de **Winnowing** no emplea algún tipo de cifrado, por lo que al igual que el proceso chaff no tiene una "clave de descifrado". Intentando regular la confidencialidad que provee un cifrado damos paso a la esteganografía y el proceso de winnowing.[20]

Existen dos partes en el envío de mensajes con winnowing: Autentificación (Agregando MACs) y agregando paquetes chaff. Nosotros nos enfocaremos más al uso de paquetes chaff para el envío seguro de información, ya que, el receptor es quien remueve los paquetes chaff para obtener el mensaje original.

Los siguientes esquemas explican como es que se lleva a cabo el proceso de **Chaffing and Winnowing** en diferentes escenarios.

**Escenario 1:** Alice se está comunicando con Bob en un solo camino de comunicación sobre un canal inseguro y Charles agrega los paquetes de Chaff.

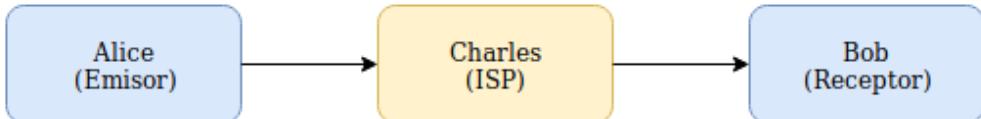


Figura 2.3: Charles agrega los paquetes inválidos.

En el escenario anterior Alice y Bob se están comunicando mutuamente por un canal de comunicación no seguro, en donde son enviados paquetes no cifrados. Alice y Bob comparten la llave de autentificación la cual será usada para el proceso de autentificación. Cuando Alice envía un mensaje a Bob, su mensaje es autenticado de su lado y es enviado a Charles antes de ser enviado a Bob. Charles agrega los paquetes chaff a la secuencia transmitida por Alice, al agregar los paquetes chaff, Charles provee confidencialidad para la comunicación entre Alice y Bob. Pero donde Charles no conoce la llave secreta compartida entre Alice y Bob. Por lo que el proceso de chaffing no necesita ningún conocimiento de la llave secreta de autentificación compartida.

**Escenario 2:** Alice se comunica con Bob en un camino de comunicación inseguro y en el cual Charles no agrega los paquetes chaff si no que multiplexa los flujos de las otras dos partes (David y Elaine). Este escenario es diferente al anterior, ya que se multiplexó el flujo de datos de Alice y Bob con el flujo de datos de David y Jane, y cuando el paquete llega a Bob el flujo de paquetes de David hacia Jane es el chaff de Bob y es descartado y vice versa para Jane.

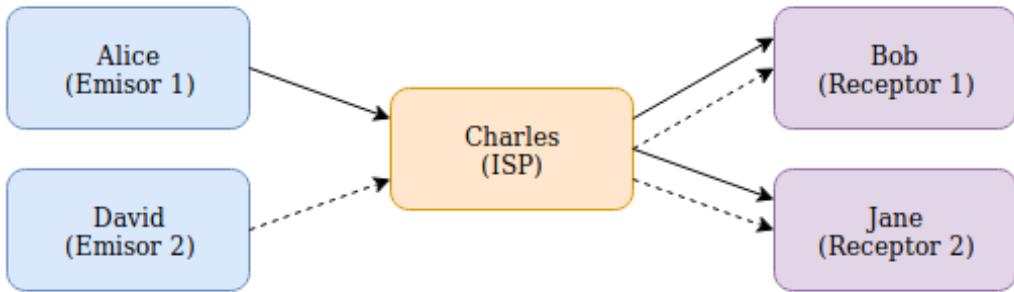


Figura 2.4: Charles no agrega los paquetes pero multiplexa los flujos.

**Escenario 3:** Alicia se comunica con Bob en un canal de comunicación inseguro y Alice no agrega los paquetes chaff. En este escenario, Alice desarrolla la autentificación de sus mensajes, por lo que Alice aplica chaffing para autenticar los mensajes y producir una secuencia de paquetes que serán transmitidos a Bob por la vía de Charles. Bob lleva a cabo el proceso de winnowing para recuperar el mensaje original.

### 4.3. Objetivo de Chaffing and Winnowing

El objetivo de seguridad del esquema de chaffing-and-winnowing es proporcionar privacidad en un entorno simétrico. Desde un punto de vista de seguridad, este esquema debe tratarse simplemente como un esquema de cifrado simétrico. Hay algunos procesos de "cifrado" que toman un mensaje y crean un "texto cifrado", y algún proceso de "descifrado" toma el texto cifrado y recupera el mensaje, ambos operando bajo una clave secreta en común. (Para el esquema Chaffing and Winnowing es la clave para el MAC). Estos procesos no se implementan de manera "habitual", pero, de manera abstracta, deben existir, de lo contrario no se logra la privacidad. [18] [19]

*"No es una propiedad de seguridad novedosa, sino un conjunto novedoso de restricciones en los procesos dirigidos a lograr una propiedad de seguridad estándar"*

*"Encontrar-luego-adivinar". Extensión más directa al caso simétrico de la noción de indistinguibilidad.*

Haciendo uso de **Chaffing and Winnowing** se asegura que los adversarios no obtengan información del mensaje transmitido a lo largo de un canal de comunicación inseguro entre dos partes.

Rivest propone un esquema, el cual cuenta con tres partes principales. [17]

1. **Autentificación** Es el proceso de descomponer el mensaje original en un paquete más pequeño y complementar cada paquete con un código de autentificación de mensaje (MAC).
2. **Chaffing** Es el proceso de agregar paquetes inválidos (Chaff packets).
3. **Winnowing** Es el proceso de remover paquetes Chaff para obtener el mensaje original en texto plano.

#### 4.4. ¿Cómo funciona?

El esquema de Chaffing and Winnowing deja que cada paquete conste de:

- Un número de serie
- Contenido del paquete
- Código de autentificación del mensaje

Cuando son enviados los paquetes, el mensaje con el texto plano se descompone en pequeños paquetes los cuales contienen datos y el tamaño del paquete original. Entonces, el emisor (Alice) usa el algoritmo de **código de autentificación de mensaje** (MAC) para generar el valor MAC para ser agregado al paquete y el cual se basa en el número de serie, contenido del paquete y la llave autentificación. A continuación se muestra la salida del paquete después del proceso de autentificación.

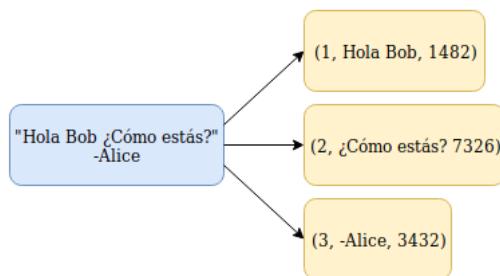


Figura 2.5: Secuencia de Chaffing después del proceso de autentificación.

Esta secuencia de paquetes es enviada a Charles (ISP) para llevar a cabo el proceso de Chaffing. Charles agrega paquetes chaff a la secuencia de paquetes antes de ser enviados por medio del canal de comunicación y ser recibidos por Bob.

Existen dos maneras donde Charles puede enviar la secuencia de chaff hacia Bob. La primera es enviando aleatoriamente mezclados los paquetes chaff para formar una secuencia y la otra manera es enviarlos de manera ordenada por el número de serie seguido del contenido del mensaje. En la siguiente figura se muestra cómo es el proceso de chaff en esta secuencia.

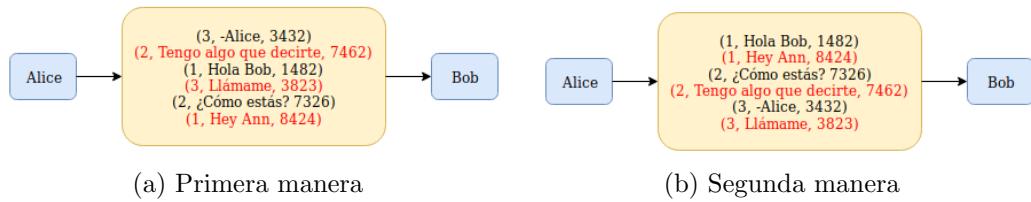


Figura 2.6: Las dos maneras para el proceso de chaff pueden ser utilizadas. Los paquetes chaff son los mensajes de color rojo.

Una vez que la secuencia de chaff llega a Bob, el último proceso es Winnowing. Bob determina la secuencia del mensaje que es válida del paquete chaff usando una función hash para el contenido de cada paquete y la llave de autentificación para re-calcular el MAC y compararlo contra el MAC del paquete recibido, si la comparación falla, el paquete chaff es descartado. Si la comparación es valida, entonces el paquete es parte del mensaje original. La siguiente imagen muestra el proceso completo de chaffing and winnowing.

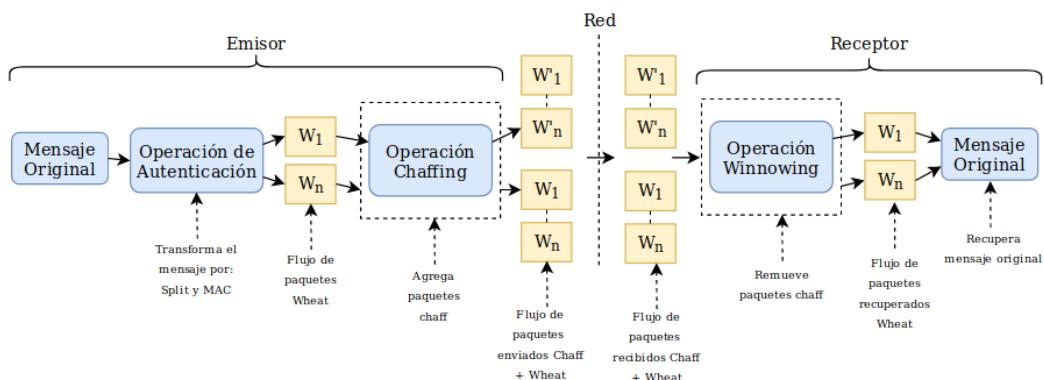


Figura 2.7: Visión general del proceso Chaffing and Winnowing.

## 4.5. Propiedades de Chaffing and Winnowing

- La técnica de Chaffing y Winnowing no depende de la fuerza del esquema de cifrado para proporcionar confidencialidad debido al hecho de que es muy difícil distinguir la información útil de los paquetes chaff sin la clave secreta. Por lo tanto, la dificultad de distinguir la información útil del chaff proporciona confidencialidad al esquema.
- La operación de Chaffing puede ser realizada por un tercero, ya que la clave secreta compartida no es necesaria en el proceso del mismo.
- Los paquetes de Chaff no tienen que contener datos aleatorios, ya que uno podría usar un mensaje válido con una clave secreta diferente para hacer el paquete de Chaff. Cuando el receptor recibe esos paquetes de Chaff, se verán como paquetes de Chaff, ya que la clave que se usa para volver a calcular el Chaff es diferente de la que los hace.

#### 4.6. All-or-Nothing and the Package Transform (AONT)

All-or-Nothing and the Package Transform es una variación dentro de la técnica Chaffing and Winnowing, donde se mejora la eficiencia de su esquema original. AONT es la transformación de pre-procesamiento que permite a las partes enviar más datos (en términos de bit) por paquete en lugar de solo uno. Este pre-procesamiento es una transformación sin cifrado que toma el mensaje de texto sin formato y produce un mensaje empaquetado que luego se procesa de la manera normal de Chaffing and Winnowing. Las definiciones de la transformación AONT son las siguientes:

1. El algoritmo de transformación es **reversible**: Dado el bloque de mensaje transformado, el receptor puede obtener el mensaje de texto sin formato original.
2. El algoritmo de transformación y su inverso son **computables** de manera eficiente: Lo que significa que es computacionalmente factible recrear el texto original dada la llave privada y recibir todos los paquetes con éxito.
3. La transformación no es **computacionalmente factible**: Esto significa que si se ha recibido parte del paquete de la transmisión, cualquiera que esté intentando leer el mensaje no puede hacerlo ya que la transformación **AONT** requiere que se reciba todo el mensaje, de lo contrario no entrega nada.
4. La transformación es una **técnica sin cifrado**: La técnica de pre-procesamiento no tiene llaves y no hay una llave secreta compartida

involucrada en la operación. Cualquier persona que haya recibido todos los mensajes transformados del paquete puede recuperar el mensaje de texto original.

### *¿Cómo funciona AONT?*

Supongamos que el mensaje de entrada es el siguiente:  $m_1, m_2, \dots, m_n$ . Seleccionamos una llave aleatoria  $K'$  el cual se usará para la función del paquete de transformación. Se calcula la secuencia transformada  $m'_1, m'_2, \dots, m'_s$  para  $s' = s + 1$  como se muestra a continuación:

Tenemos:

$$m_i \otimes E(K', i) \text{ for } i = 1, 2, 3, \dots, s$$

También:

$$m'_{s'} = K' \otimes h_1 \otimes h_2 \otimes \dots \otimes h_s$$

Donde:

$$h_i = E(K_0, m'_i \otimes i) \text{ for } i = 1, 2, \dots, s$$

Donde  $K_0$  es una llave conocida pública fija.

Para que el receptor en el otro extremo obtenga el  $K_0$ , el cual es la llave para el uso de **AONT**, el receptor realiza el siguiente cálculo:

$$K' = m'_s \otimes h_1 \otimes h_2 \otimes \dots \otimes h_s$$

$$m_i = m'_i \otimes E(K', i) \text{ for } i = 1, 2, \dots, s$$

AONT toma el mensaje de texto sin formato de entrada y los transforma, luego crea un bloque para almacenar los mensajes transformados antes de pasar al proceso de autentificación. Después, se genera el paquete Chaff (la cantidad de paquetes Chaff no tiene que ser igual a los paquetes de la información útil).

Esta técnica produce una menor sobrecarga que la sugerencia número 1. El AONT ofrece más confidencialidad al esquema de Chaffing and Winnowing, ya que el adversario debe recibir todo el bloque de mensajes de transformación e identificar correctamente todo el paquete de la información útil para obtener el mensaje de texto original. La siguiente figura muestra la descripción general de Chaffing y Winnowing si se agrega la función AONT.

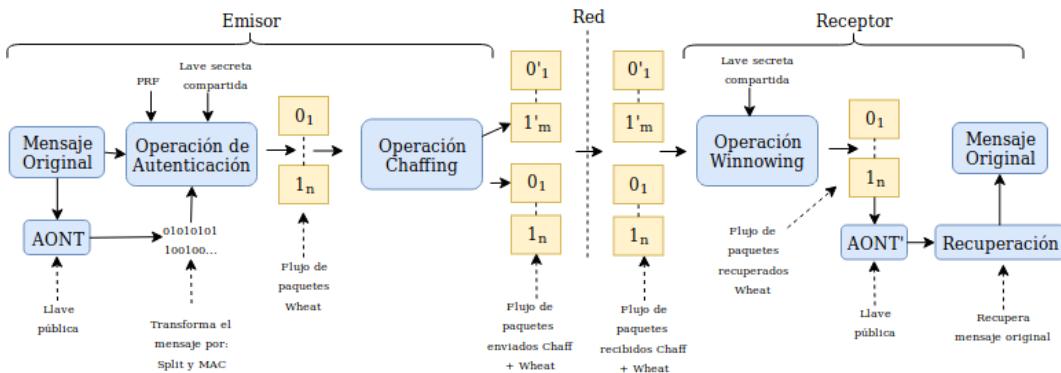


Figura 2.8: Proceso de Chaffing and Winnowing junto con AONT.

### ¿Cómo AONT puede hacer la diferencia?

1. Requiere menos ancho de banda al transferir paquetes, ya que se pueden transferir más bits en un paquete en lugar de un bit por paquete.
2. Los paquetes Chaff son más fáciles de generar, ya que AONT transforma el mensaje de texto plano en bits aleatorios.
3. La distinción entre Chaffing and Winnowing es más difícil: Si el adversario va a ejercer fuerza bruta en los paquetes, la tarea se ralentizará por el factor del número de bloque de mensajes. Dado que el bloque de mensaje de información adicional se mezcla aleatoriamente dentro de los flujos de paquetes de Chaffing and Winnowing, sin saber que es muy difícil que el bloque adicional proporcione la posibilidad de elegir el bloque de mensaje correcto de los paquetes para obtener el texto plano original.

## 4.7. Comparando Chaffing and Winnowing contra Cifrado y Esteganografía

En esta sección explicaremos porque Chaffing and Winnowing no puede ser clasificado como una técnica de cifrado o Estenografía.

### 4.7.1. Chaffing and Winnowing vs Cifrado

Nosotros podríamos clasificar Chaffing and Winnowing como un método de cifrado, pero volvamos a recordar el principio de un Cifrado. El principal

objetivo de un cifrado es ocultar el mensaje en texto plano de tal manera que oculta su contenido con el uso de una clave de cifrado para el texto cifrado. Por otro lado, en el esquema original de Chaffing and Winnowing, una llave compartida es usada con el fin de autenticar la validación de los paquetes ya sea del emisor o del receptor. Además, Chaffing and Winnowing no hace uso de ninguna técnica de cifrado para ocultar el contenido de un mensaje y que nadie pueda ver dicho mensaje, solo aquellos con la llave correspondiente pueden determinar que paquetes contienen la información valida. En la figura 2.8, se muestra como se puede ver el esquema Chaffing and Winnowing como una técnica de cifrado.

Chaffing y Winnowing pueden verse como **un tipo especial de esquema de cifrado simétrico**, ya que la operación **chaffing** es similar al "proceso de cifrado". En la operación de chaffing, el texto cifrado se crea para producir un paquete de la información útil no válido que se envía al receptor. Luego, el receptor realiza el "proceso de descifrado", que implica descartar el paquete de desperdicios y recuperar el mensaje original. Ambas operaciones operan bajo una llave secreta común que se usa para derivar el valor MAC.

Pero la diferencia es, *Chaffing y Winnowing* dos partes que no buscan lograr la confidencialidad. El emisor comparte una clave secreta con el receptor para que el receptor pueda usar la clave secreta para autenticarse (si se afirma que el mensaje recibido proviene del remitente deseado). Pero la ganancia de confidencialidad proviene de la dificultad de distinguir el paquete Chaff del paquete de la información útil. Mientras que en el cifrado, la clave se utiliza para lograr la confidencialidad mediante la creación de texto cifrado que oculta el contenido del mensaje de personas.

Chaffing y Winnowing junto con el esquema AONT, el esquema en sí es muy parecido al cifrado, excepto que la clave que se usa en la transformación AONT, se elige aleatoriamente cada vez en lugar de fijarla. Además, el último bloque de mensajes es exclusivo o de la clave y todo el hash del bloque de mensajes está allí para garantizar que cualquier modificación en el bloque de mensajes cambiará la clave  $K'$  calculado por el receptor. Por lo tanto, el último bloque de mensajes  $m'_{s'}$  está allí solo con el propósito de autenticación. Por lo tanto, Chaffing y Winnowing con el esquema AONT no pueden ser clasificados bajo cifrado.

#### **4.7.2. Chaffing and Winnowing vs Esteganografía**

Para algunas personas, Chaffing and Winnowing puede ser clasificado como una técnica esteganografía. Sin embargo, el objetivo principal de la esteganografía es el de ocultar el mensaje original dentro de otro tipo de mensaje, por lo tanto, nadie aparte del emisor y el receptor sabrá que hay un mensaje oculto. Contrario a Chaffing and Winnowing, en donde cualquiera puede ver el contenido del mensaje, ya que este método no trata de esconderlo de los posibles atacantes. Otra diferencia es que en esteganografía el emisor tiene que ocultar el mensaje el mismo, mientras que en Chaffing and Winnowing no necesariamente es así, ya que una "tercera parte" puede hacerlo.

Por lo tanto, Chaffing and Winnowing no puede ser considerado como esteganografía.

Para nuestro proyecto usaremos *Chaffing and Winnowing*, para proponer un nuevo método de autentificación para servicios web.

# **Capítulo 3**

## **Análisis.**

### **1. Estudio de Factibilidad.**

El estudio de factibilidad es un instrumento que sirve para orientar la toma de decisiones en la evaluación de un proyecto y corresponde a la última fase de la etapa pre-operativa dentro del ciclo del proyecto. Se formula con base en información que tiene la menor incertidumbre posible para medir las posibilidades de éxito o fracaso de un proyecto, apoyándose en él se tomará la decisión de proceder o no con su implementación. Este estudio establecerá la viabilidad, si existe, del proyecto.

- Factibilidad Técnica: Hace referencia a los recursos como herramientas, conocimientos, habilidades, experiencia, etc. que son necesarios para efectuar las actividades del proyecto.
- Factibilidad Operativa: Se refiere a los recursos necesarios para llevar a cabo los procesos de forma eficiente , depende de los recursos humanos.
- Factibilidad Económica: Consiste en los recursos financieros necesarios para llevar a cabo la elaboración de este proyecto.

#### **1.1. Factibilidad Técnica**

En esta parte explicaremos detalladamente las tecnologías que usaremos. Para la elección de estas herramientas fue necesario investigar las tecnologías que más se usan en la actualidad, además de ver las características y equipos de cómputo con los que contamos actualmente.

Factibilidad Técnica	
Sistema Operativo	Multiplataforma
Navegador Web	Google Chrome
Lenguaje de Programación	JavaScript
Servidor	Apache 2.0

Cuadro 3.1: Herramientas de Software a utilizar

Además de las herramientas de software a utilizar, es necesario mencionar el equipo de hardware que utilizaremos tanto para desarrollar como para probar e implementar cada uno de los prototipos que se mencionarán a lo largo de este trabajo terminar, el cual es:

Equipo de hardware [1]	
Marca	DELL
Modelo	Inspiron 5567
Procesador	Intel Core i7 7gen
Tarjeta de video	Radeon (TM) R7 M445
Memoria RAM	16 GB
Disco Duro	1 TB

Cuadro 3.2: Equipo de hardware a utilizar [1]

Equipo de hardware [2]	
Marca	Asus
Modelo	X550VC
Procesador	Intel Core i5
Tarjeta de video	NVidia GForce 720
Memoria RAM	12 GB
Disco Duro	1 TB

Cuadro 3.3: Equipo de hardware a utilizar [2]

Equipo de hardware [3]	
Marca	HP
Modelo	Pavilion g4
Procesador	Intel Core i3
Tarjeta de video	Intel Sandybridge Mobile
Memoria RAM	6 GB
Disco Duro	500 GB

Cuadro 3.4: Equipo de hardware a utilizar [3]

Junto con las herramientas de hardware y software a utilizar es necesario mencionar una serie de servicios básicos que son relevantes para el desarrollo de este proyecto como lo son

- Luz Eléctrica
- Agua Potable
- Internet
- Papelería en general

Estos servicios forman parte de la factibilidad técnica ya que sin ellos no se podría realizar este proyecto y por eso mismo generan un costo, dicho costo se menciona en la Factibilidad Económica.

## 1.2. Factibilidad Operativa

Los recursos operativos de este proyecto se calcularon con base en los recursos humanos con los que se cuenta y un análisis de las horas que el personal estará en operación trabajando sobre este, el cual se muestra a continuación:

Horas a trabajar en el desarrollo del proyecto						
Mes	No. de Días	Sábado y Domingo	Días hábiles	Horas de trabajo por día	Horas Totales	—Días laborables (8 hr.)
Enero	31	8	9	2	18	—2
Febrero	28	8	19	2	38	—4
Marzo	31	10	20	2	40	—5
Abril	30	10	15	2	30	—3
Mayo	31	8	18	2	36	—4
Junio	30	10	8	2	16	—2
Agosto	31	9	12	2	24	—3
Septiembre	30	10	16	2	32	—4
Octubre	31	10	20	2	40	—5
Noviembre	31	8	18	2	36	—4

Cuadro 3.5: Relación de horas de trabajo estimadas para la realización de este proyecto

Con esto podemos concluir que contamos suficiente tiempo para el desarrollo de este proyecto, ya que las horas totales de trabajo están contempladas para cada uno de los integrantes del equipo

### 1.3. Factibilidad Económica

Luego de haber realizado el estudio de factibilidad técnica así como el operacional es necesario tomar en cuenta un estudio de factibilidad económica el cuan desglosará todo el gasto económico realizado para la elaboración de este proyecto:

- Capital Humano: Se tienen contemplados aproximadamente 36 días laborales, es decir 288 horas para la elaboración de este proyecto en el cual participaremos los cuatro integrantes
- Capital Técnico: Se cuentan con las instalaciones de la escuela, así como las viviendas de cada uno de los integrantes y los equipos de cómputo correspondientes.

En cuanto a los costos monetarios de todo el proyecto se tiene lo siguiente:

- Servicios  
En cuando a los servicios se considera un gasto mensual aproximado

de \$1,600.00 que al multiplicarlo por todo el tiempo de elaboración tenemos \$ 16,000.00.

- Software

En este caso durante todo el proyecto usaremos herramientas gratuitas y la mayoría de software libre por lo que no dedicaremos una parte monetaria en el gasto de este tipo.

- Hardware

En este caso y como se mencionó anteriormente utilizaremos los equipos de cómputo personales de cada integrante lo que da un costo aproximado total de \$ 35,000.00.

- Recursos Humanos

Estamos estimando un gasto de \$80,000.00 por cada integrante para la elaboración de este proyecto por lo que se genera un gasto total de \$320,000.00

Por lo que el costo final del desarrollo de este proyecto es:

\$371,000.00

**Conclusión** Tras analizar todo este proyecto y cada una de las partes del estudio de factibilidad es pertinente decir que los integrantes no contarán con el apoyo financiero antes mencionado y que el hardware actualmente ya es propiedad de los integrantes, por lo que el proyecto se califica como "**Viable**" iniciando de esta manera su implementación acorde con las fechas mencionadas.

## 2. Herramientas a usar.

### 2.1. Software.

Para el desarrollo de software de este prototipo, es necesario hacer mención de algunas de las siguientes herramientas, para tener una idea clara sobre qué herramientas estamos utilizando y porque es que las estamos utilizando:

**HTML5.** HTML comenzó mucho tiempo atrás con una simple versión propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información, todo esto tenía la intención de comunicar información por medio de texto. El limitado objetivo de html motivó a varias

compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas.

Dos de las opciones propuestas fueron Java y Flash; ambas fueron muy aceptadas y consideradas como el objetivo de la internet, sin embargo, con el crecimiento exponencial del internet, éste dejó de ser únicamente para los aficionados de los computadores y pasó a ser usado como un campo estratégico para los negocios y para la interacción social, ciertas limitaciones presentes en ambas tecnologías probaron ser una sentencia de muerte. Esta falta de integración resultó ser crítica y preparó el camino para la evaluación de un lenguaje del cual hablaremos un poco más a detalle después: JavaScript. Sin embargo, pese a su gran impacto, el mercado no terminó de adoptarlo plenamente y rápidamente su popularidad fue declinando, y el mercado terminó enfocando su atención a Flash. No fue hasta que los navegadores mejoraron su intérprete para JavaScript y la gente se empezaba a dar cuenta de las limitaciones que ofrecía Flash, que JavaScript fue implementado y comenzó a innovar la forma en la que se programaba la web. Al cabo de unos años, JavaScript, html y css eran considerados como la más perfecta combinación para evolucionar la Web.

HTML5 es una mejora de esta combinación, lo que unió todos estos elementos. HTML5 propone estándares para cada aspecto de la Web y también un propósito claro para cada una de las tecnologías involucradas. A partir de esto, html provee los elementos estructurales, CSS se concentra en volver esta estructura utilizable y atractiva a la vista, y JavaScript tiene todo lo necesario para brindar dinamismo y construir aplicaciones web completamente funcionales. Cabe mencionar que HTML5 funciona diferente dependiendo del navegador y la versión en la que se esté trabajando, algunos soportan más características o diferentes funcionalidades que otros.

### CSS3.

Ya se ha mencionado anteriormente como es que HTML5 fue evolucionando a un grado de combinación de estructura y diseño, sin embargo, la web demanda diseño y funcionalidad, no solamente organización estructural o definición de secciones, la función de CSS se concentra en volver la estructura de HTML utilizable y atractivo a la vista.

Oficialmente CSS no tiene nada que ver con HTML4, no es parte de la especificación, es de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al principio, atributos den-

tro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que HTML evolucionó, la escritura de códigos se volvió más compleja y html por sí mismo no pudo satisfacer más las demandas de los diseñadores. En consecuencia a esta demanda, CSS fue adoptado como la forma de separar la estructura de la presentación, y ha ido creciendo y ganando importancia, pero siempre desarrollado en paralelo enfocado en las necesidades de los diseñadores y apartado de la estructura de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño, Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas. Las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5. [8]

### **JavaScript.**

JavaScript es considerado como el lenguaje de programación de html y de la web. Es un lenguaje de programación fácil de usar y muy versátil para el ámbito de la comunicación en redes. Los programas, llamados "scripts", se ejecutan en el navegador (Mozilla, Google Chrome, Internet Explorer, etc.) normalmente consisten en unas funciones que son llamadas desde el propio html cuando algún evento sucede.

Su primera aproximación a un uso real, fue en mayor parte para "dar vida a una página web", como dar animaciones a un botón, interacciones en tiempo real, entre otras más. JavaScript fue desarrollado por Netcape, a partir del lenguaje Java, que en ese momento tenía mucho auge y popularidad, y su principal diferencia es que JavaScript sólo "funciona" dentro de una página html.

JavaScript fue declarado como estándar del European Computer Manufacturers Association (ECMA) en 1997, y poco después, también fue estandarizado por ISO.[7]

JavaScript es un lenguaje interpretado, usado mayormente como complemento de ciertos objetivos específicos, sin embargo, uno de las innovaciones que ayudó a JavaScript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento del código. La clave de los motores

más exitosos fue transformar el código de Javascript en código máquina para obtener una velocidad de ejecución mejor que antes. Esto a la vez permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje JavaScript como la mejor opción para la Web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, JavaScript fue expandido en cuestión de portabilidad e integración, a la vez, interfaces de programación de aplicaciones (APIs) fueron incorporando por defecto con cada navegador para asistir a JavaScript en funciones elementales. El objetivo de esto, fue principalmente hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la Web.<sup>[8]</sup>

### CryptoJs.

CryptoJs es una colección estándar y segura de algoritmos criptográficos implementados para JavaScript usando las mejores prácticas y patrones. Es rápida y tiene una interfaz consistente y simple. Cuenta con algoritmos tales como: Hasher Algorithms (SHA-1, SHA-2), Cipher Algorithms (DES, AES), Block Modes and Padding, por mencionar algunos. [25] El objetivo de usar esta herramienta es usar los métodos ya implementados en *CryptoJs* para la generación de código hash para una mejor y rápida ejecución del sistema.

### Algoritmo MD5.

El algoritmo de **hashing** criptográfico *MD5* es una versión mejorada de su antecesor, MD4. Las siglas MD corresponden a *Message Digest* o resumen de mensaje y fueron diseñados por *Ron Rivest*. El algoritmo produce, a partir de una entrada no limitada en cuanto a su tamaño, un *hash* criptográfico, o resumen de mensaje, de 128 bits de longitud. Se lo emplea actualmente en diferentes aplicaciones de seguridad, una de las más populares es la comprobación de integridad de archivos. Estas comprobaciones suelen representarse en una cadena de 32 caracteres alfanuméricos, que corresponden a los 16 bytes en formato hexadecimal. [6]

El uso del algoritmo de hashing se empleará en la creación del certificado para un usuario (tomando como parámetros el nombre de usuario y su contraseña), debido a que en este prototipo aún no se cuenta con el *servidor autenticador*. El mensaje como resultado de aplicar el algoritmo hash, se inyectará en el protocolo HTTP, como se ha mencionado en la descripción

del prototipo II.

### **Wireshark.**

Wireshark es una analizador de paquetes de red. Un analizador captura paquetes de red y muestra los datos del paquete con el mayor detalle posible. Esta herramienta es software libre y es el mejor analizador de paquetes hoy en día. [26] Utilizaremos Wireshark para analizar los paquetes que viajan a través de la red, en este caso la petición modificada la cual contiene el certificado y el patrón ocultados mediante el método *Chaffing*.

## **2.2. Hardware.**

En el ámbito del hardware, utilizaremos los equipos de cómputo con los cuales contamos actualmente los integrantes de este equipo, los cuales se especificarán a continuación:

Equipo de hardware utilizado.	
Marca	Asus
Modelo	X550VC
Procesador	Intel Core i5
Tarjeta de video	NVidia GForce 720
Memoria RAM	12 GB
Disco duro	1TB

Equipo de hardware utilizado.	
Marca	HP
Modelo	Pavilion g4
Procesador	Intel Core i3
Tarjeta de video	Intel Sandybridge Mobile
Memoria RAM	6 GB
Disco duro	500GB

Equipo de hardware utilizado.	
Marca	DELL
Modelo	Inspiron 5567
Procesador	Intel Core i7
Tarjeta de video	Radeon (TM) R7 M445
Memoria RAM	16 GB
Disco duro	1TB

Equipo de hardware utilizado.	
Marca	HP
Modelo	Pavilion 15-p000ns
Procesador	AMD A A8-5545M
Tarjeta de video	Radeon R8 M540
Memoria RAM	8 GB
Disco duro	1TB

### 3. Arquitectura del sistema.

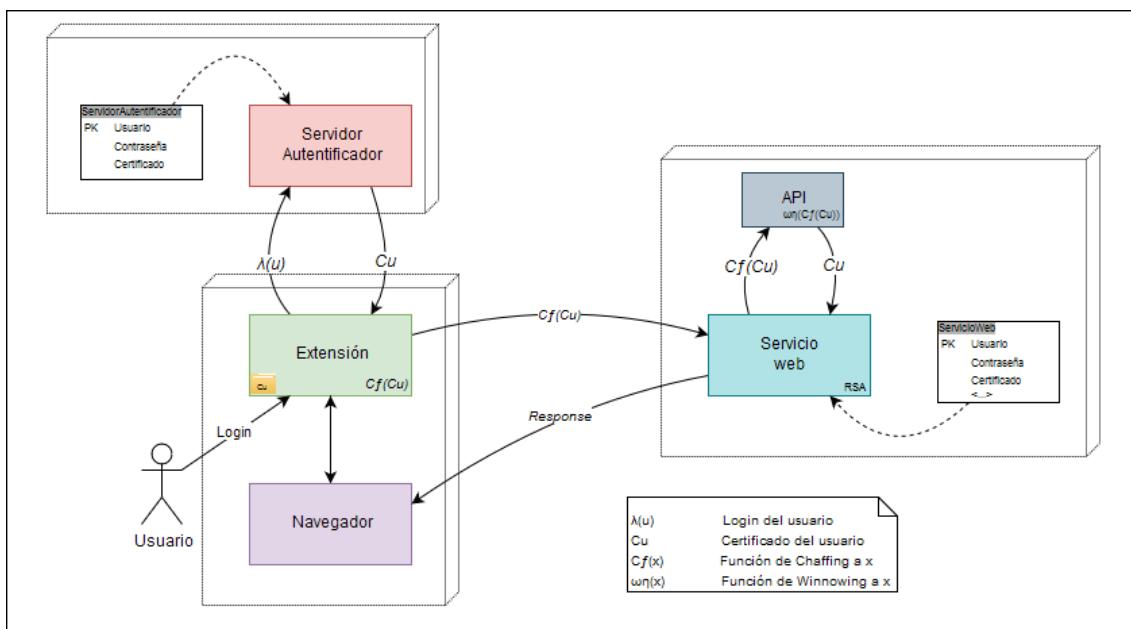


Figura 3.1: Arquitectura General del Sistema

## 4. Diagrama de casos de uso general.

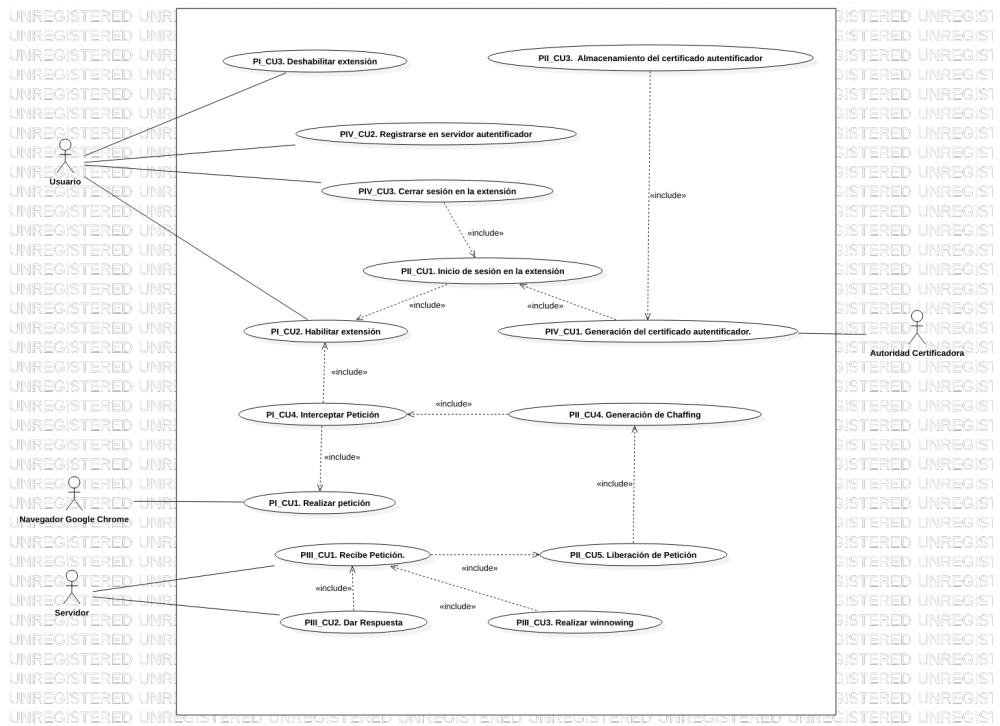


Figura 3.2: Diagrama de casos de uso general del sistema

NOTA: en la sección de desarrolla, cada prototipo describirá sus casos de uso correspondientes para su funcionamiento.

## 5. Prototipo I.

### 5.1. Descripción.

En este prototipo se busca la creación de una extensión de Google Chrome que pueda interceptar una petición HTTP hecha por el mismo navegador mientras se encuentre habilitada, y así evitar que dicha petición sea mandada al servidor. Además, la extensión mostrará en otra pestaña del navegador información sobre la petición interceptada, con la finalidad de analizar los datos del encabezado HTTP.

El propósito de realizar este prototipo es familiarizarse con el manejo de extensiones en el navegador Google Chrome; como es que podemos obtener la información que necesitamos para que posteriormente modifiquemos esta petición y la envíemos al servidor de prueba.

### 5.2. Estudio de requerimientos.

#### 5.2.1. Requerimientos Funcionales.

**PI\_RF1. Interceptar petición HTTP.** La extensión deberá interceptar la petición HTTP del navegador, en cuanto el usuario realice alguna a través de éste.

**PI\_RF2. Deshabilitar extensión.** El usuario podrá deshabilitar la extensión, para que ésta no vigile su actividad en el navegador.

**PI\_RF3. Habilitar extensión.** El usuario podrá habilitar la extensión, para que ésta vigile las peticiones HTTP.

**PI\_RF4. Validar petición.** La extensión deberá analizar la petición previamente recibida, y validar si ésta es HTTP(S) o no.

**PI\_RF5. Mostrar petición.** La extensión deberá mostrar en otra pestaña del navegador, la información de la petición que se haya realizado.

**PI\_RF6. Bloquear salida de petición.** La extensión deberá evitar que la petición salga a red, deteniéndola cuando se le haya aplicado la función *Chaffing* en el Prototipo II.

### **5.2.2. Requerimientos no Funcionales.**

**PI\_RNF1. Plataforma de implementación.** La extensión será implementada en el navegador Google Chrome Desktop.

**PI\_RNF2. Versión del navegador** La extensión funcionará a partir de la versión 28.0.

**PI\_RNF3. Tecnologías para la interfaz de usuario** Para el sistema se hará uso de html5, JavaScript, css3, JSON.

**PI\_RNF4. Permitir ejecución de JavaScript en Google Chrome.** Para el correcto funcionamiento de la extensión, es necesario que se permita la ejecución de javascript en el navegador Google Chrome.

**PI\_RNF5. Conexión a internet.** Para el funcionamiento de la extensión, no es necesario que se tenga conexión a internet.

### **5.3. Reglas del negocio.**

**PI\_RN1. Extensión habilitada.** En cuanto el usuario lo indique por medio de la Interfaz de Usuario, la extensión deberá vigilar la actividad que éste realice en el navegador para interceptar una petición.

**PI\_RN2. Extensión deshabilitada.** En cuanto el usuario lo indique por medio de la Interfaz de Usuario, la extensión deberá dejar de vigilar la actividad que éste realice en el navegador.

## **6. Prototipo II.**

### **6.1. Descripción.**

En este prototipo se busca que la extensión de Google Chrome pueda modificar la petición HTTP previamente interceptada. La modificación se hará sólo mientras la extensión esté habilitada, y tiene como objetivo injectar el certificado autentificador en el encabezado del protocolo. Este certificado será simulado (sólo para este prototipo) por la misma extensión a partir de los datos del inicio de sesión. Para la creación de dicho certificado, se usará el algoritmo de hashing MD5, tomando como argumentos de entrada usuario y contraseña, generando así, una cadena resultante de 128 bits, sin embargo, para los siguientes prototipos, se manejará una autoridad certificadora la cual recibirá los datos del inicio de sesión para que genere el certificado del usuario. Una vez que dicho certificado es injectado, la extensión deberá liberar la petición para que salga a red.

El propósito de este prototipo es utilizar la técnica de *Chaffing and Winnowing* en este nuevo método de autenticación, para evitarle al usuario la tediosa tarea de ingresar sus credenciales cada vez que accede al servicio y brindarle la seguridad necesaria al iniciar de sesión.

Para injectar el certificado autentificador, es necesario crear un "*patrón de chaffing*", este patrón lo generaremos aleatoriamente para después mandarlo junto con la petición HTTP. Por el momento, dicho patrón no irá cifrado, ya que para esto es necesario la implementación del servidor del servicio web de prueba, para poder conocer su llave pública y cifrar el patrón. Sin embargo, en el prototipo III se implementará el servidor y con ello su cifrado asimétrico. El objetivo de mandar el patrón junto con el protocolo HTTP, es que el servidor pueda leer el patrón y realizar la etapa de *winnowing*, para extraer el certificado.

Los algoritmos necesarios para hacer el *chaffing* son expuestos a continuación:

```

Data: requestHTTP, cert
Result: patternChaffing
1 lenHTTP  $\leftarrow$  lenght(requestHTTP);
2 lenCert  $\leftarrow$  lenght(cert);
3 lenPattern  $\leftarrow$  lenHTTP + lenCert;
4 patternChaffing[lenPattern];
5 count  $\leftarrow$  0;
6 y  $\leftarrow$  lenHTTP % 8;
7 while count < lenCert do
8   | x  $\leftarrow$  random % lenPattern;
9   | if x <= 0 && x < 8 - y && y not 0 then
10  |   | continue;
11  | end
12  | if patternChaffing[x] == 0 then
13  |   | patternChaffing[x]  $\leftarrow$  1;
14  |   | count  $\leftarrow$  count + 1;
15  | end
16 end
17 byteControl  $\leftarrow$  8 - y;
18 putFirst(patternChaffing, byteControl);

```

**Algorithm 1:** Generación de patrón de chaffing

**Data:** requestHTTP, cert, patternChaffing  
**Result:** protocolChaffed

```

1 lenPattern  $\leftarrow$  lenght(PatternChaffing);
2 protocolChaffed[lenPattern] countHTTP  $\leftarrow$  0;
3 countCert  $\leftarrow$  0;
4 controlByte  $\leftarrow$  getFirstByte(patternChaffing);
5 entero  $\leftarrow$  byteToInt(controlByte);
6 countProtocolChaffed  $\leftarrow$  entero;
7 while countProtocolChaffed  $<$  lenPattern do
8   if protocolChaffed[countProtocolChaffed]  $=$  0 then
9     protocolChaffed[countProtocolChaffed]  $\leftarrow$  cert[countCert];
10    countCert  $\leftarrow$  countCert + 1;
11  else
12    protocolChaffed[countProtocolChaffed] =
13      requestHTTP[countHTTP];
14    countHTTP  $\leftarrow$  countHTTP + 1;
15  end
16  countProtocolChaffed  $\leftarrow$  countProtocolChaffed + 1;
17 end
18 randomShift  $\leftarrow$  rdShift();
19 binaryCicleRightShiftByByte(protocolChaffed, randomShift);
20 putFirst(patternChaffing, randomShift);

```

**Algorithm 2:** Generación del chaffing

Primero vamos a analizar el algoritmo del *patrón de chaffing* con un pequeño ejemplo (utilizaremos datos de variables más cortos); supongamos que nuestro certificado es el siguiente:

$$C_k = MITZ057abZ251$$

y utilizaremos el campo *agent user* del header de la petición, lo cual tendrá lo siguiente:

$$P_{HTTP} = Mozilla5,5/Chrome8,1/Safari$$

Entonces, nuestro patrón de chaffing terminará teniendo un tamaño de la longitud de los caracteres de nuestro certificado + la longitud de los datos de la petición HTTP, que es nuestra variable *lenPattern*, y en este caso será de una longitud de 41. A continuación vamos a utilizar un arreglo de banderas de ese tamaño que inicia con 0 todos sus valores, y que al final será nuestro patrón, (variable *patternChaffing*).

La idea es generar posiciones aleatorias dentro del rango del tamaño de *lenPattern*, y poner un 1 en dicha posición si es que hay un 0, en caso de que no se encuentre un 0 se repetirá el procedimiento obteniendo una posición aleatoria diferente, y esto se realizará el mismo número de veces del tamaño de nuestro certificado, para obtener un patrón de combinaciones entre los caracteres. Utilizaremos un contador para conseguir esto, aumentándolo cada vez que se obtenga un número aleatorio válido. En nuestro algoritmo, nuestra variable *x* contiene la posición aleatoria donde se validará si en esa posición se puede poner un 1 o no.

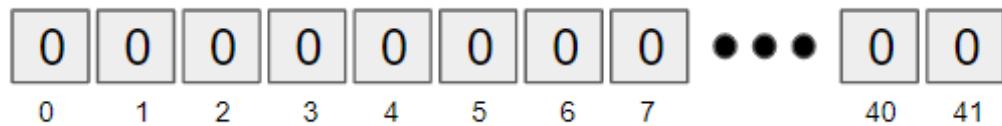


Figura 3.3: Arreglo del patrón de chaffing

Supongamos, por ejemplo, que el primer número aleatorio que se obtiene es 4, entonces en la posición 4 colocaremos un 1:



Figura 3.4: Arreglo del patrón de chaffing después de una iteración

Este algoritmo nos permite llenar de unos el número de veces del tamaño de nuestro certificado, lo que nos asegura que cada posición le corresponde a un carácter ya sea del certificado o de la petición a enviar. Supongamos que al terminar este algoritmo, nuestro arreglo queda algo parecido a lo siguiente:



Figura 3.5: Arreglo del patrón de chaffing final

El segundo algoritmo realizará el chaff utilizando el patrón generado en el algoritmo anterior, obtendrá el byte de control y a partir de este, comenzará un contador para recorrer todos los bytes del patrón de chaffing, ademas contaremos con otros 2 contadores más: *contCert* que irá recorriendo cada carácter de nuestro certificado, y *countHTTP* que recorrerá los de la petición HTTP. Si en la posición de éste contador se encuentra un 0, entonces quiere decir que debemos de agarrar el siguiente carácter de nuestro certificado, basándonos en el contador del certificado, de la misma manera si se encuentra un 1, se agarrará el siguiente carácter de la petición, y estos se irán concatenando en una nueva cadena (*protocolChuffed*). Al final, para incrementar la seguridad de nuestro *chaff*, se realizará un corrimiento que se ve en la función de *binaryCircleRightShiftByByte*<sup>1</sup>.

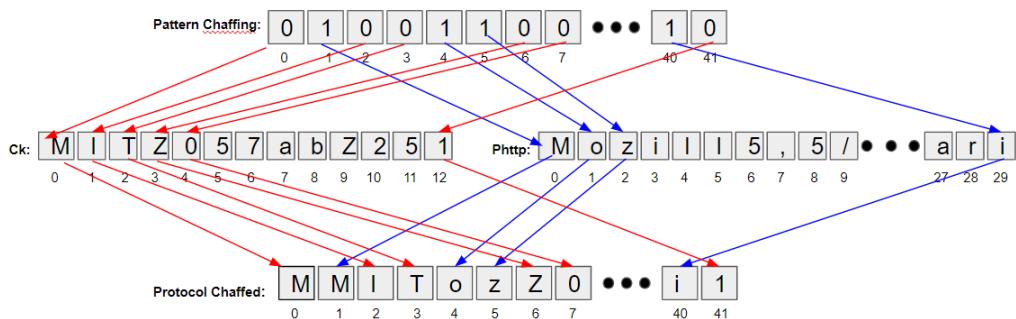


Figura 3.6: Arreglo Chaff final.

Al final necesitaremos un byte de control que le servirá al servidor saber a partir de donde empezará a realizar el *winnowing*, esto se debe a que dependiendo del tamaño de la petición HTTP y su módulo 8 es donde empezará a checar los 0's y 1's; mandaremos este byte de control al principio de la petición con chaff.

Por último se enviará al servidor el patrón de chaffing junto con el mismo chaff, así podrá saber como realizar el *winnowing* para obtener los datos correspondientes del certificado.

---

<sup>1</sup>Este corrimiento será cíclico y basado en el código ASCII, es decir, se escogerá el carácter de n bytes a su derecha módulo de 256.

## 6.2. Estudio de requerimientos.

### 6.2.1. Requerimientos Funcionales.

**PII\_RF1. Inicio de sesión en la extensión.** La extensión contará con una interfaz para el ingreso de datos del usuario, donde ingresará un usuario y contraseña.

**PII\_RF2. Generación del certificado autentificador.** Para el prototipo 2 la generación del certificado autentificador se hará del lado del cliente, simulando la tarea de una entidad certificadora, permitiendo así generar automáticamente esta llave la cual servirá para el proceso de *chaffing* del mensaje en el protocolo HTTP. Dicho certificado se generará con base a los datos del PII\_RF1. Inicio de sesión.

**PII\_RF3. Almacenamiento del certificado autentificador.** La extensión deberá almacenar el certificado autentificador.

**PII\_RF4. Generación de patrón de Chaffing.** La extensión generará un patrón para poder implementar el método de *Chaffing*. Este patrón será generado al azar, y es aquel que se usará para introducir el código autentificador en el protocolo HTTP.

**PII\_RF5. Etapa de Chaffing.** Por medio del método *Chaffing* se crearán paquetes para agregar al encabezado HTTP que se ha interceptado gracias al prototipo 1, utilizando el patrón de *Chaffing* del requerimiento funcional PII\_RF4. Generación de patrón de *Chaffing*

**PII\_RF6. Liberación de Petición.** Se liberará el bloqueo a la petición HTTP impuesto por el prototipo I una vez terminado el proceso de *Chaffing*, para que la petición pueda llegar al servidor modificado correspondiente.

### 6.2.2. Requerimientos no Funcionales.

**PII\_RNF1. Tamaño del código autentificador.** El tamaño del archivo depende totalmente del tamaño de llave que se desea generar. Para este prototipo se hará uso de la función *hash* tomando como argumentos el nombre de usuario y contraseña del usuario que se desea generar el código de hash.

**PII\_RNF2. Almacenado de archivo en la extensión.** Se necesita tener cargado el archivo en la API de Google Chrome, **Storage** para el correcto

funcionamiento de la función *Chaff* y así inyectarlo en la petición.

**PII\_RNF3. Conexión a internet.** Para el funcionamiento de este prototipo, es necesario tener acceso a internet.

### **6.3. Reglas del negocio.**

**PII\_RN1. Petición válida.** La extensión modificará la petición siempre y cuando se trate de una petición válida HTTP .

**PII\_RN2. Extensión habilitada.** La modificación de la petición HTTP solo se podrá realizar mientras la extensión se encuentre habilitada.

**PII\_RN3. Inicio de sesión de extensión por usuario.** Cada usuario que desee utilizar la extensión sólo deberá tener una cuenta con un usuario y una contraseña respectiva a este usuario.

**PII\_RN4. Acceso a internet.** Se debe de contar con acceso a internet para que la extensión pueda enviar al servidor la petición modificada.

**PII\_RN5. Longitud de código autentificador.** La longitud del código autentificador es de 32 caracteres, debido a que la función de MD5 es un algoritmo de codificación de 128 bits que genera un hash hexadecimal de 32 caracteres, independientemente de la longitud de la palabra de entrada (nombre de usuario y contraseña).

**PII\_RN6. Longitud del campo de usuario.** La longitud del usuario no debe pasar de 20 caracteres, y mínimo será de 3 caracteres.

**PII\_RN7. Longitud del campo contraseña.** La longitud de la contraseña no debe pasar de 16 caracteres, y mínimo sera de 8 caracteres.

**PII\_RN8. Caracteres permitidos en campo usuario.** Los caracteres permitidos en el campo de usuario son únicamente símbolos alfanúmericos y guion bajo.

**PII\_RN9. Caracteres permitidos en campo contraseña.** Los caracteres permitidos en el campo de contraseña son únicamente símbolos alfanúmericos así como los símbolos \_ @ / # ? .

**PII\_RN10. Formato de contraseña.** La contraseña debe tener al menos un número y una letra mayúscula.

## 7. Prototipo III.

### 7.1. Descripción.

Para este prototipo, vamos a implementar un servicio web de prueba para realizar la etapa de *winnowing* a la petición HTTP que le llegue. Esto implica la modificación del servidor para que detecte este nuevo tipo de autenticación y la creación de un algoritmo que realice el *winnowing* al protocolo. El servidor que usaremos será Apache.

Así mismo, el servicio web implementará un cifrado asimétrico (RSA), esto con la finalidad de que la extensión del prototipo anterior, pueda tomar la "llave pública" del servidor y cifrar el "*patrón de chaffing*" que se mandará en la petición, para que sólo el servidor pueda descifrar este patrón con su "llave privada". El propósito de este prototipo es poder obtener el certificado autenticador, para poder validarla y dar acceso o no al usuario. Cabe mencionar que, la primera vez que el servidor reciba el certificado autenticador, pedirá que se "inicie sesión" con la finalidad de poder asociar el certificado a una cuenta; el inicio de sesión en el servicio web, sólo se hará una vez, ya que para las peticiones posteriores, el certificado ya tendrá una cuenta asociada a la cual dar acceso.

## 8. Prototipo IV.

### 8.1. Descripción.

Para este prototipo, vamos a implementar un servidor autenticador en el cual se crearán y almacenarán los certificados de los usuarios. En este servicio, los usuarios tendrán registrada una cuenta a la cual accederán desde la extensión de los prototipos mencionados anteriormente. Una vez que inicien sesión (usuario y contraseña), este servidor autenticador regresará como respuesta el certificado generado para que la extensión lo administre.

Para la creación del certificado autenticador utilizaremos la herramienta OpenSSL, además, la comunicación entre extensión y servidor se hará bajo SSL/TLS.

El propósito de este prototipo es poder crear y utilizar un certificado real, el cual sea capaz de autenticar a un usuario en un servicio web.

# Capítulo 4

## Desarrollo.

### 1. Prototipo I.

#### 1.1. Diagrama de casos de uso.

Diagrama de casos de uso general para el prototipo I.

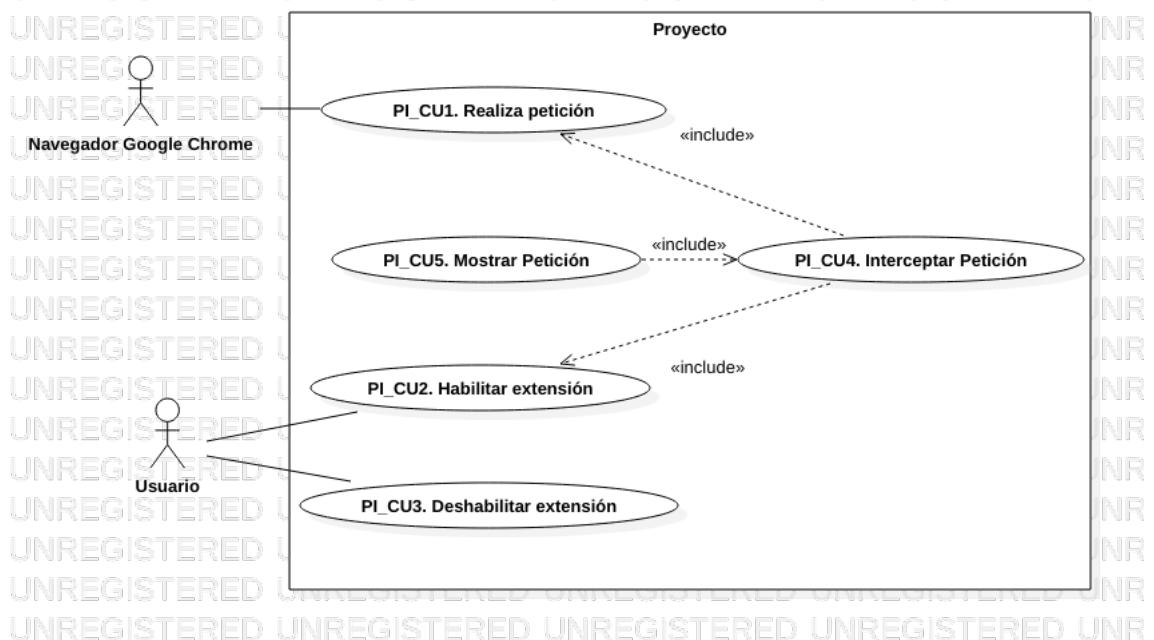


Figura 4.1: Diagrama de casos de uso del Prototipo I.

## 1.2. Descripción de casos de uso.

Caso de uso: PI_CU1. Realizar petición.	
Concepto	Descripción
Actor	Navegador de Google Chrome.
Propósito	Este caso de uso permite al navegador realizar una petición HTTP, ordenada por el usuario o un sistema externo.
Entradas	URL del servicio web solicitado.
Salidas	Petición HTTP.
Pre-condiciones	Algún agente externo (Sistema o usuario) ha ordenado al navegador mandar una petición HTTP.
Post-condiciones	Creación de la petición HTTP.
Reglas del negocio	-
Errores	La petición no se pudo realizar. La petición no es tipo HTTP.

Cuadro 4.1: Descripción CU: PI\_CU1

... Trayectoria Principal ...

1. **El Usuario o El Sistema Externo** realiza una petición HTTP en el navegador Google Chrome.
2. **El navegador** realiza la petición.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. **El Usuario o El Sistema Externo** realiza una petición que no es HTTP en el navegador Google Chrome.
2. **El navegador** realiza la petición.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: PI_CU2. Habilitar extensión.	
Concepto	Descripción
Actor	Usuario.
Propósito	Este caso de uso, permite al usuario habilitar la extensión, para que ésta sea capaz de ver todas las peticiones que realiza el navegador.
Entradas	Indicación de habilitar extensión, mediante interfaz de usuario.
Salidas	-
Pre-condiciones	Estar en la <b>UI servicio desactivado. Figura 4.8</b>
Post-condiciones	-
Reglas del negocio	<b>PI_RN1. Extensión habilitada.</b>
Errores	No se puede habilitar la extensión.

Cuadro 4.2: Descripción CU: PI\_CU2

... Trayectoria Principal ...

1. *El usuario* da click en el ícono de la extensión .
2. *El usuario* da click en el botón .
3. *La extensión* empieza a vigilar las peticiones que se realicen a través del navegador.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *La extensión* no muestra el botón , por ende el usuario no puede dar click.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: PI_CU3. Deshabilitar extensión.	
Concepto	Descripción
Actor	Usuario.
Propósito	Este caso de uso permite al usuario deshabilitar la extensión, para que ésta ignore todas las peticiones que se realicen por medio del navegador.
Entradas	Indicación de deshabilitar extensión, mediante interfaz de usuario.
Salidas	-
Pre-condiciones	Estar en la <b>UI servicio activado. Figura 4.7.</b>
Post-condiciones	-
Reglas del negocio	<b>PI_RN2. Extensión deshabilitada</b>
Errores	No se puede deshabilitar la extensión.

Cuadro 4.3: Descripción CU: PI\_CU3

... Trayectoria Principal ...

1. *El usuario* da click en el ícono de la extensión .
2. *El usuario* da click en el botón **Desactivar**.
3. *La extensión* deja de vigilar las peticiones que se realicen a través del navegador.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *La extensión* no muestra el botón **Desactivar**, por ende el usuario no puede dar click.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: PI_CU4. Interceptar petición.	
Concepto	Descripción
Actor	-.
Propósito	Este caso de uso interceptará las peticiones HTTP que el navegador realice para su análisis.
Entradas	Petición realizada por el navegador
Salidas	-
Pre-condiciones	<b>PI_CU1. Realizar petición</b> <b>PI_CU2. Habilitar extensión</b>
Post-condiciones	<b>PI_CU5. Mostrar Petición</b>
Reglas del negocio	<b>PI_RN1. Extensión habilitada</b>
Errores	La petición no se genera correctamente.

Cuadro 4.4: Descripción CU: PI\_CU4

### ... Trayectoria Principal ...

1. *El usuario* ingresa una URL en el navegador.
2. *El navegador* genera una petición HTTP.
3. *La extensión* Intercepta la petición realizada por el navegador para su análisis

### ... Fin de la Trayectoria Principal ...

Caso de uso: PI_CU5. Mostrar Petición.	
Concepto	Descripción
Actor	-
Propósito	Este caso de uso permite que la extensión cree una nueva pestaña para mostrar la petición interceptada.
Entradas	Petición HTTP creada por el navegador.
Salidas	Petición mostrada en una pestaña nueva.
Pre-condiciones	Caso de uso <b>PI_CU4. Interceptar Petición.</b>
Post-condiciones	-
Reglas del negocio	-
Errores	No se puede crear una nueva pestaña. No se puede recuperar la petición HTTP. No se puede imprimir la petición en la nueva pestaña.

Cuadro 4.5: Descripción CU: PI\_CU5

... Trayectoria Principal ...

1. *La extensión* recibe la petición HTTP a imprimir.
2. *La extensión* abre una nueva pestaña.
3. *La extensión* imprime la petición en UI nueva pestaña con impresión del encabezado HTTP.

... Fin de la Trayectoria Principal ...

### 1.3. Diagrama de flujo (DF).

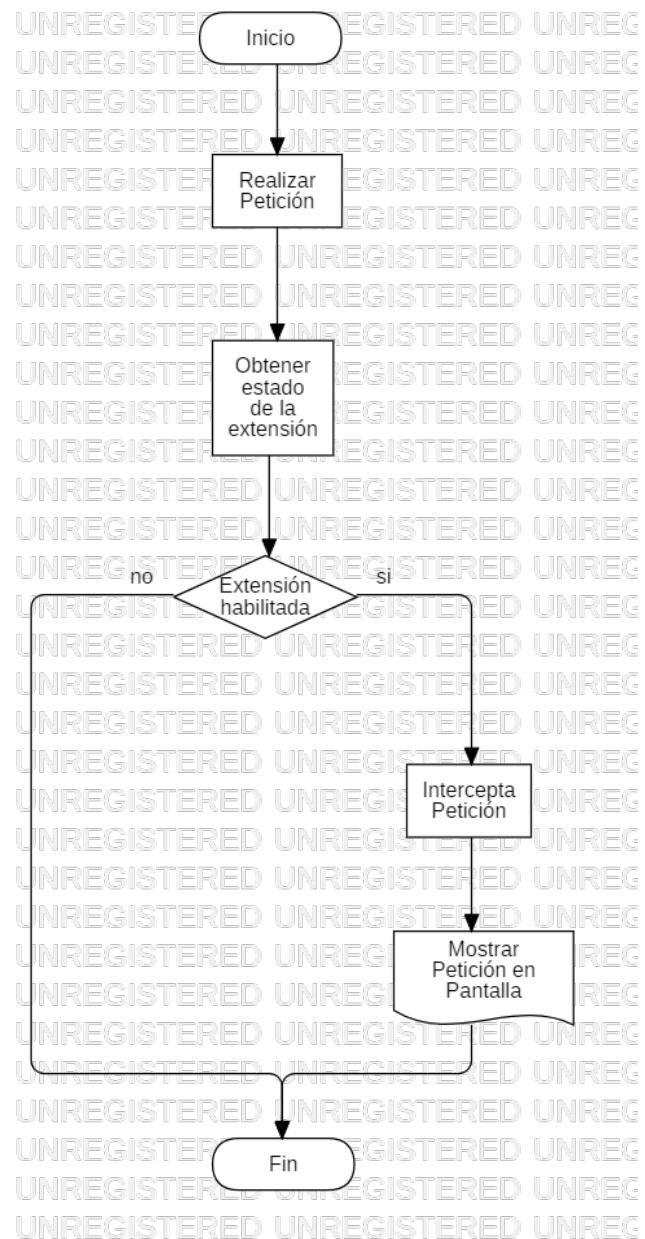


Figura 4.2: Diagrama de flujo del Prototipo 1.

### **1.3.1. Descripción diagrama de flujo.**

El primer paso del diagrama de flujo es esperar a que el usuario realice una petición web a algún servidor, luego obtener el estado actual de la extensión para saber si se encuentra habilitada o no, para luego realizar la comprobación correspondiente, es decir, “.Extensión habilitada”, en caso de que no se encuentre habilitada se acaba el flujo y ya no se realiza ninguna acción, en caso de que si se encuentre habilitada se intercepta la petición HTTP y se evita que salga de la extensión, para este entonces se crea una nueva pestaña y se muestra toda la información de la petición, hasta este punto damos por concluido el prototipo 1.

## 1.4. Diagrama de flujo de datos (DFD).

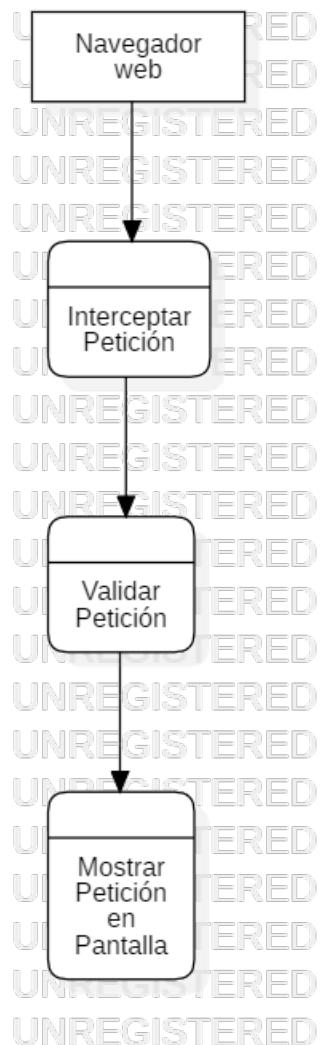


Figura 4.3: Diagrama de flujo de datos del Prototipo 1.

### 1.4.1. Descripción diagrama de flujo de datos.

Para este diagrama solo nos limitamos a los datos que se utilizan en el prototipo uno los cuales son las peticiones tanto de salida como las de entrada, al inicio la extensión intercepta la petición, los cuales son datos que se validan para detectar cuales son HTTP y después de esto simplemente se muestran los datos en pantalla.

## 1.5. Diagrama de clases.

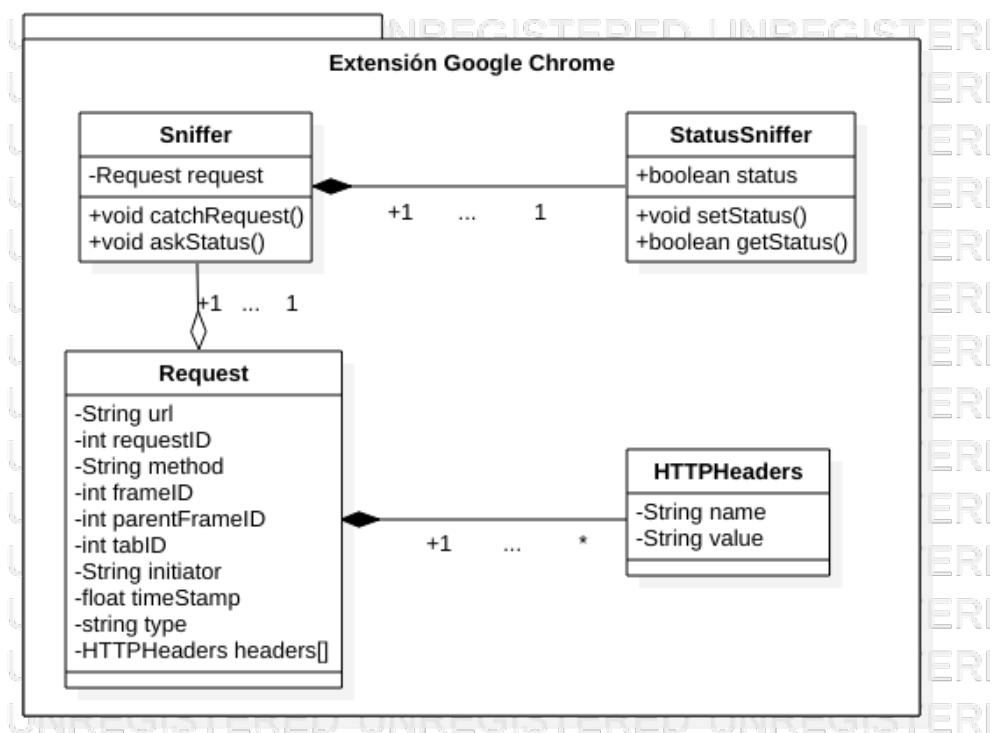


Figura 4.4: Diagrama de clases del Prototipo I.

### 1.5.1. Descripción diagrama de clases.

#### StatusSniffer

1. **status** : Variable que permite conocer y almacenar el status de la extensión (Habilitada/Deshabilitada).

a) Tipo de dato: **boolean**.

#### Métodos

a) **setStatus()**: Se establece el valor del status, True (Está activada la extensión) o False (No está activa la extensión).

- b) **getStatus()**: Se obtiene el valor del status de la extensión, True o False.

Sniffer

- 1. **request** : Variable que almacena los valores de la cabecera HTTP.

- a) Tipo de dato: **Request**.

Métodos

- a) **catchRequest()**: Intercepta la petición para obtener los valores de la cabecera.
  - b) **askStatus()**: Invoca al método *getStatus()* de la clase StatusSniffer para conocer el status de la extensión.

Request

- 1. **url** : Variable que almacena la URL requerida.

- a) Tipo de dato: **String**.

- 2. **requestID** : Variable que almacena el ID de la petición.

- a) Tipo de dato: **int**.

- 3. **method** : Variable que almacena el tipo de método de la petición, *GET* o *POST*.

- a) Tipo de dato: **String**.

- 4. **frameID** : Variable que almacena el tipo del frame.

- a) Tipo de dato: **int**.

- 5. **parentFrameID** : Variable que almacena el id del frame que envía la petición.

- a) Tipo de dato: **int**.

6. **tabID** : Variable que almacena el ID del tab que toma lugar en la petición.
  - a) Tipo de dato: **int**.
7. **initiator** : Variable que almacena el origen de la petición.
  - a) Tipo de dato: **String**.
8. **timeStamp** : Variable que almacena la hora cuando se dispara la petición.
  - a) Tipo de dato: **float**.
9. **type** : Variable que almacena el tipo de la petición.
  - a) Tipo de dato: **String**.
10. **headers** : Variable de tipo Array que almacena los headers a ser enviados junto con la petición.
  - a) Tipo de dato: **HTTPHeaders**.

#### HTTPHeaders

1. **name** : Variable que almacena el nombre del header a mandar junto con la petición.
  - a) Tipo de dato: **String**.
2. **value** : Variable que almacena el contenido del header a enviar junto con la petición.
  - a) Tipo de dato: **String**.

## 1.6. Diagramas de secuencia.

### 1.6.1. Diagrama de secuencia 1

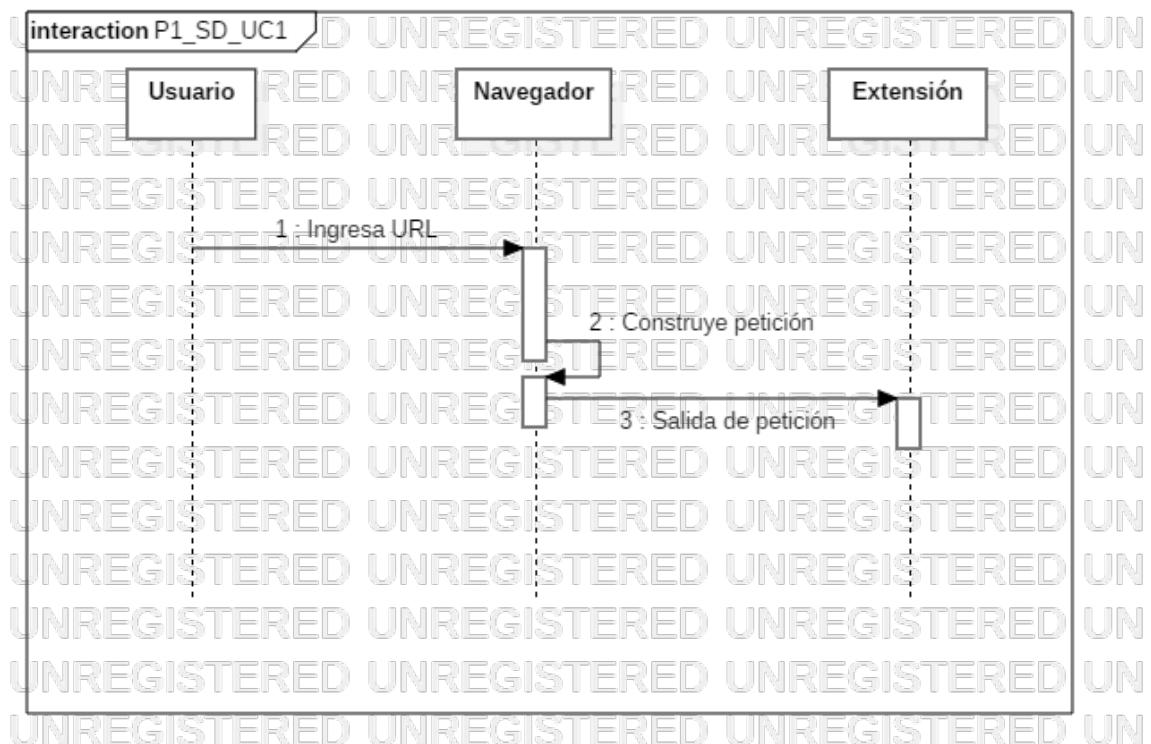


Figura 4.5: Diagrama de secuencia del PI\_CU1. Realizar petición.

### 1.6.2. Diagrama de secuencia 2

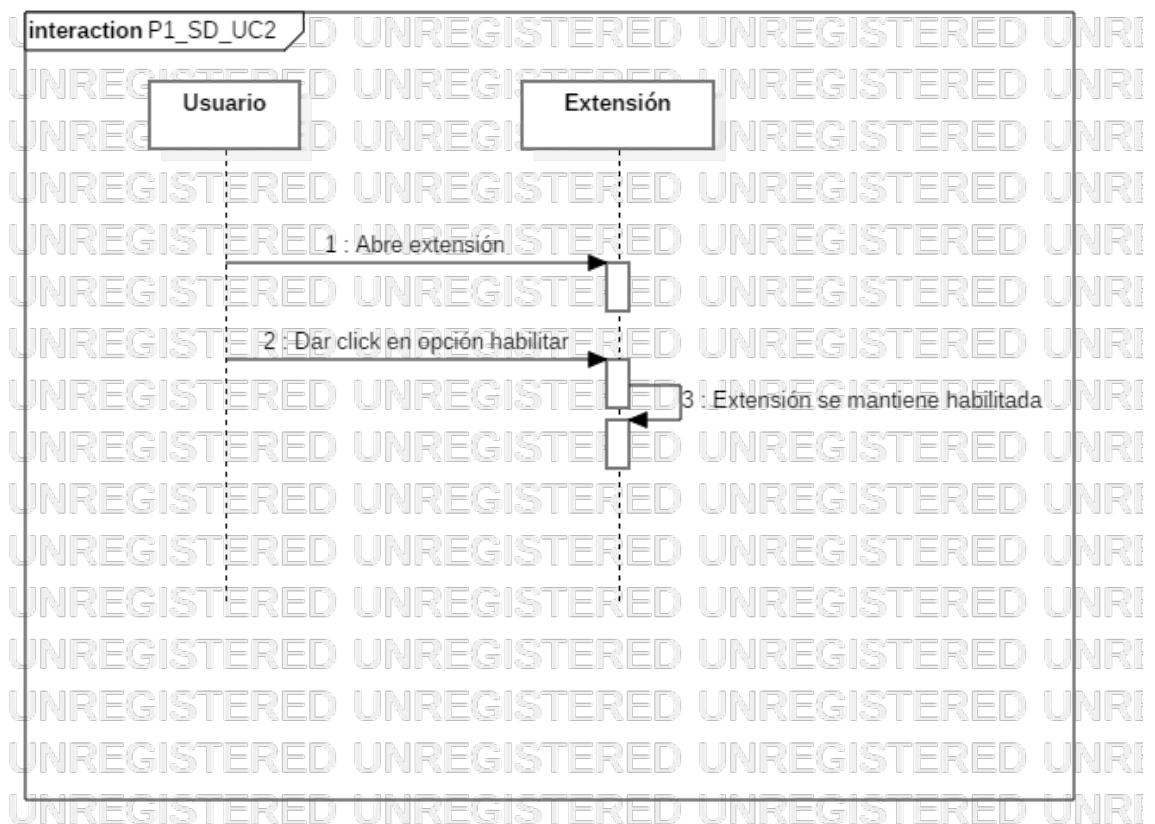


Figura 4.6: Diagrama de secuencia del PI\_CU2. Habilitar extensión.

### 1.6.3. Diagrama de secuencia 3

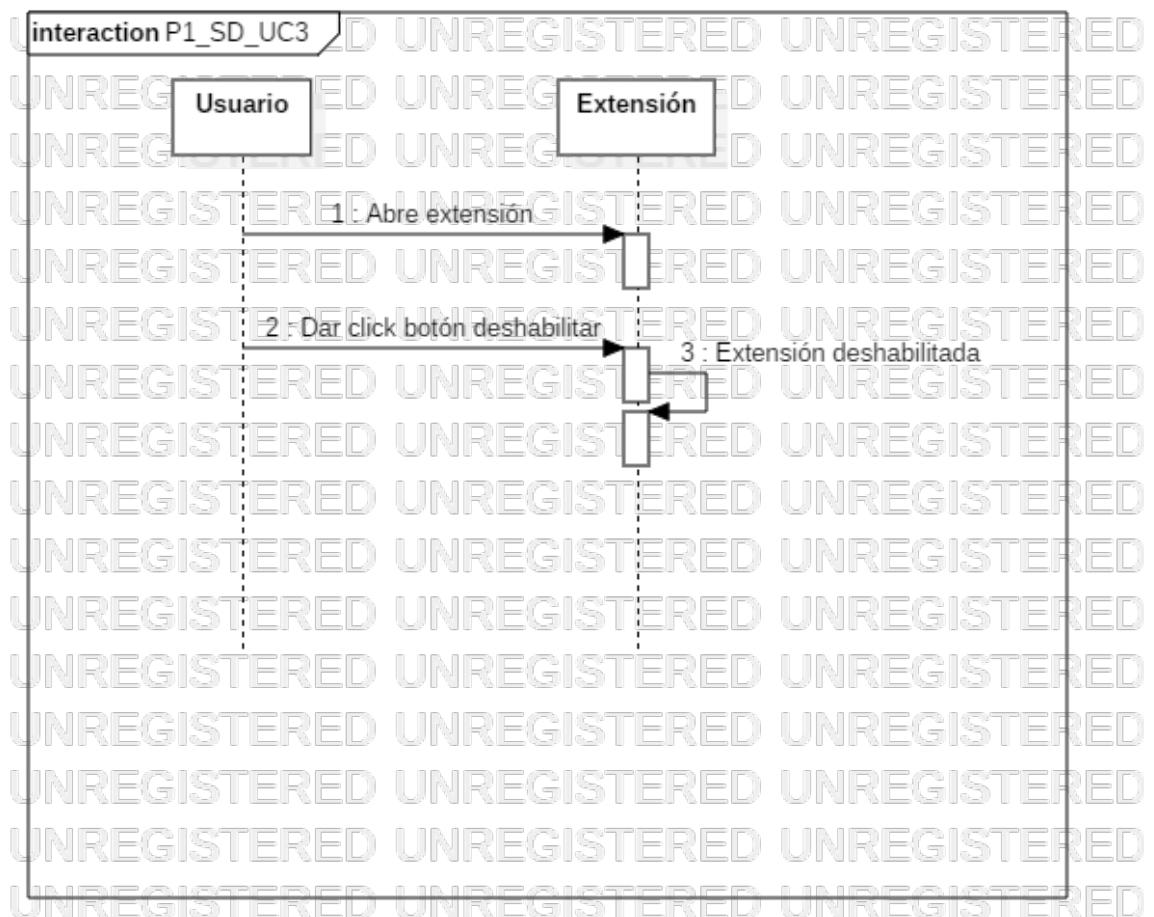


Figura 4.7: Diagrama de secuencia del PI\_CU3. Deshabilitar extensión.

#### 1.6.4. Diagrama de secuencia 4

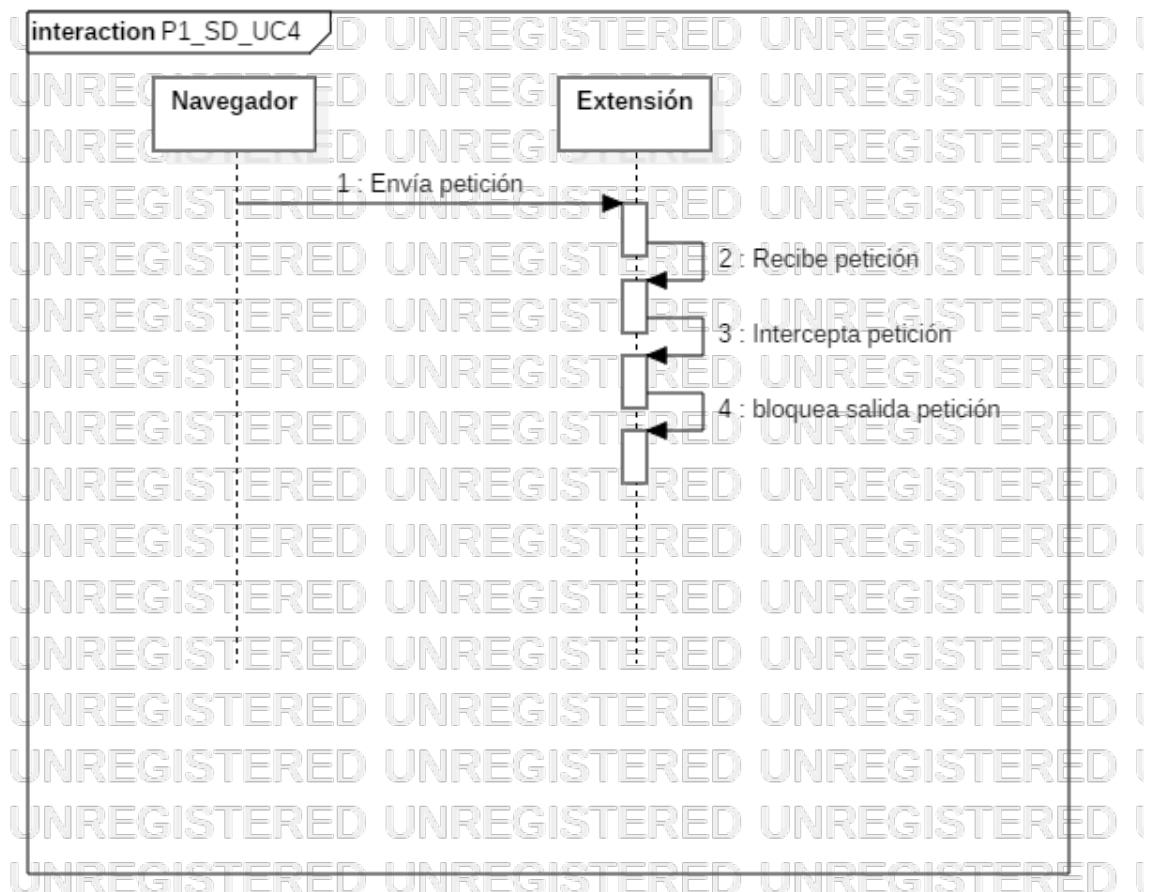


Figura 4.8: Diagrama de secuencia del PI\_CU4. Interceptar petición.

### 1.6.5. Diagrama de secuencia 5

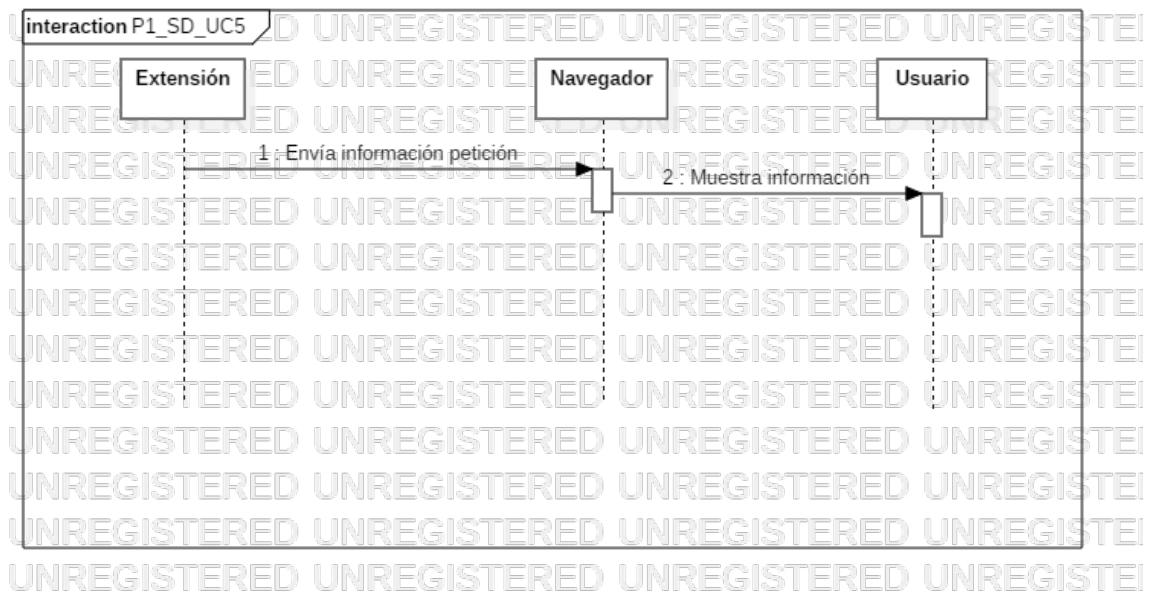


Figura 4.9: Diagrama de secuencia del PI\_CU5. Mostrar petición.

## 1.7. Diagrama de actividades

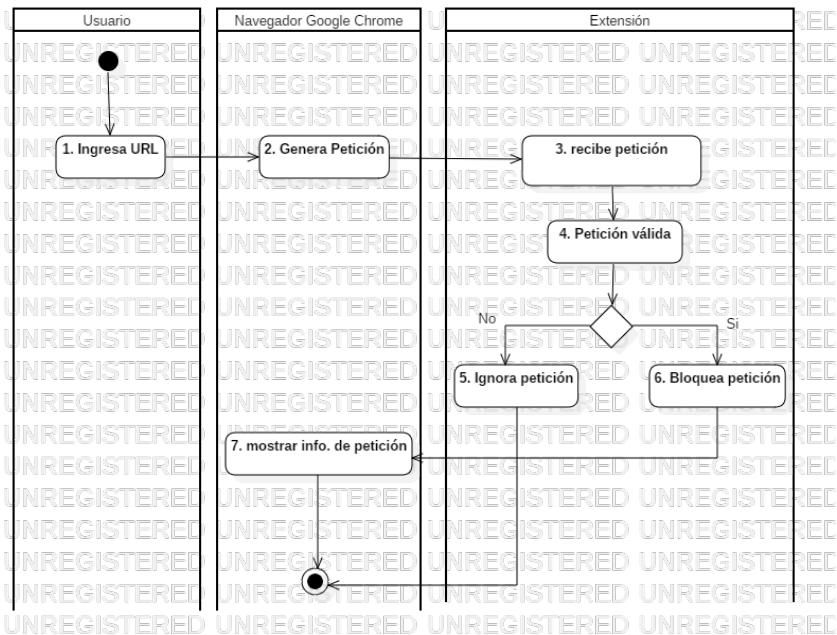


Figura 4.10: Diagrama de actividades Del Prototipo I.

### 1.7.1. Descripción de diagrama de actividades.

Este diagrama muestra las actividades importantes que toman lugar en nuestro prototipo 1, enlistadas y explicadas como siguen:

1. **Usuario Ingresa URL:** El usuario ingresa en su navegador un URL el cual el desea ingresar.
2. **Navegador genera petición:** El navegador genera una petición con base al URL que el usuario ingreso previamente.
3. **Extensión recibe petición:** La extensión recibe la petición generada

4. **Extensión valida si la petición es HTTP:** La extensión valida el tipo de petición que recibe, si es de tipo HTTP o de algún otro.
5. **Ignorar la petición:** Si la petición no es de tipo HTTP, entonces la extensión la ignora y no modifica o intercepta nada.
6. **Extensión intercepta petición:** Si la petición es de tipo HTTP, entonces la extensión debe interceptar la petición, evitando que esta salga a red y al servidor.
7. **Mostrar la petición:** Una vez interceptada la petición, la extensión muestra información sobre esta en una nueva pestaña.

### 1.8. Interfaz de usuario.



Figura 4.11: Logo de la extensión.



Figura 4.12: Pantalla inicial. Servicio activado.



Figura 4.13: Pantalla inicial. Servicio desactivado.

```

URL :: https://www.ipn.mx/
REQUEST ID :: 1106
METHOD :: GET
FRAME ID :: 0
PARENT FRAME ID :: -1
TAB ID :: 22
INITIATOR :: undefined
TIME STAMP :: 1556038938878.468
TYPE :: main_frame
HTTP[0].name :: Upgrade-Insecure-Requests
HTTP[0].value :: 1
HTTP[1].name :: User-Agent
HTTP[1].value :: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3745.4 Safari/537.36
HTTP[2].name :: Accept
HTTP[2].value :: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3

```

Figura 4.14: Nueva pestaña. Encabezado de petición.

## 1.9. Requisitos de diseño.

En este apartado, se especificarán las especificaciones de diseño para que el prototipo opere de forma correcta.

### 1.9.1. Requisitos de ejecución sobre Google Chrome.

Gracias a que Google Chrome es multiplataforma, esto es, funciona en varios sistemas operativos, el funcionamiento del Prototipo I depende exclusivamente de **tener instalado Google Chrome Stable o Google Chrome Developer** en la computadora local. Actualmente, este navegador para dispositivos móviles, no admite la instalación de extensiones, por lo que el funcionamiento depende también de tener una versión de '**Google Chrome Desktop**' instalada en la computadora. La extensión no podrá ejecutarse en Google Chrome para celulares o tabletas.

Sin embargo, para el Prototipo I utilizamos dos API's, llamadas '*chrome.webRequest*' y '*chrome.storage*', las cuales están disponible sólo a partir de la **versión 28**, por lo que, es necesario tener una versión igual o superior de este software.

Por otro lado, es **necesario que se permita que la extensión se ejecute sobre Google Chrome con ciertos permisos**,

los cuales se exponen a continuación.

- Habilitar extensión: activado
- Acceso al sitio web: en todos los sitios
- Permitir acceso a URL de archivo: activado

## 2. Prototipo II.

### 2.1. Diagrama de casos de uso

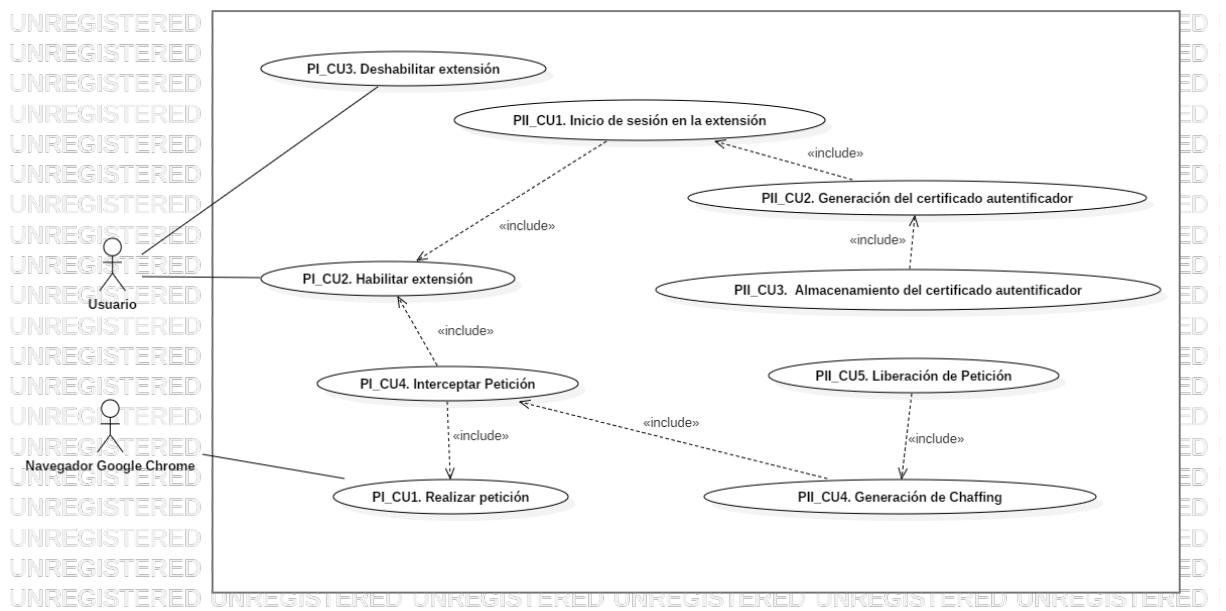


Figura 4.15: Diagrama de casos de uso del Prototipo II.

## 2.2. Descripción de casos de uso.

Caso de uso: PII_CU1. Inicio de sesión en la extensión	
Concepto	Descripción
Actor	Usuario
Propósito	<p>Este caso de uso permite al usuario poder iniciar sesión en la extensión para posteriormente obtener un código autentificador, basado en esos datos, al cual se le aplicará <i>Chaffing</i>.</p> <p>Sólo se requerirá iniciar sesión una sola vez, ya que después, sólo es necesario subir el código generado para autenticarse en el servicio web.</p>
Entradas	Usuario y Contraseña
Salidas	Código autentificador el cual hace referencia al usuario iniciado
Pre-condiciones	Haber instalado la extensión en el navegador Google Chrome y habiéndola habilitado.
Post-condiciones	Con los datos introducidos por el usuario, se obtendrá el código autentificador del mismo, para que lo guarde en su dispositivo y lo pueda subir después.
Reglas del negocio	PII_RN2 PII_RN3 PII_RN6 PII_RN7 PII_RN8 PII_RN9 PII_RN10
Errores	No se encuentra usuario registrado No se puede obtener el código autentificador No se pudo guardar el código autentificador Contraseña no válida

Cuadro 4.6: Descripción CU: PII\_CU1

... Trayectoria Principal ...

1. **El Usuario** realiza un inicio de sesión por primera vez en la extensión.
2. **La extensión** obtendrá los parametros de usuario y con-

traseña para generar un código autentificador único.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *El Usuario* ingresa usuario y/o contraseña no validos.
2. *La extensión* no podrá obtener los datos del usuario.

... Fin de la Trayectoria Alternativa 1 ...

### Caso de uso: PII\_CU2. Generación del certificado autentificador.

Concepto	Descripción
Actor	-
Propósito	Este caso de uso generará un certificado para el usuario, basados en los datos que ingrese en la extensión (usuario y contraseña).
Entradas	Usuario y contraseña ingresados en la extensión.
Salidas	Certificado autentificador del usuario.
Pre-condiciones	Contar con la extensión habilitada. e ingresar un usuario y contraseña
Post-condiciones	-
Reglas del negocio	PII_RN6. PII_RN7. PII_RN8. PII_RN9. PII_RN10.
Errores	Error al generar el certificado.

Cuadro 4.7: Descripción CU: PII\_CU2

... Trayectoria Principal ...

1. ***La extensión*** generará un código autentificador basado en los datos ingresados por el usuario.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***La extensión*** no puede generar el código autentificador.
2. ***La extensión*** mostrará al usuario un mensaje de dicho error.

... Fin de la Trayectoria Alternativa 1 ...

### Caso de uso: PII\_CU3. Almacenamiento del certificado autentificador.

Concepto	Descripción
Actor	-
Propósito	<p>Este caso de uso permite que la extensión guarde el certificado autentificador, para que el usuario no tenga que hacerlo por el mismo. De esta manera, la extensión tendrá acceso al certificado cada que éste sea requerido.</p> <p>El lugar donde se guarda dicho certificado es en "Google Chrome Storage", una zona de memoria que Google Chrome nos brinda al desarrollar una extensión para el navegador.</p>
Entradas	Certificado autentificador.
Salidas	-
Pre-condiciones	Haber terminado el caso de uso <b>PII_CU2. Generación del certificado autentificador.</b>
Post-condiciones	-
Reglas del negocio	PII_RN5. Longitud del certificado autentificador.
Errores	No se puede almacenar el certificado autentificador.

Cuadro 4.8: Descripción CU: PII\_CU3

... Trayectoria Principal ...

1. ***La extensión*** ha creado el certificado autentificador.
2. ***La extensión*** guarda el certificado en storage.
3. ***La extensión*** muestra al usuario el siguiente mensaje "Certificado guardado en Storage" en la pantalla **Figura 4.19 Certificado guardado en Storage.**

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***La extensión*** no ha creado el certificado autentificador.

2. ***La extensión*** no guarda el certificado autentificador en storage.
3. ***La extensión*** muestra al usuario el siguiente mensaje "Certificado no encontrado" en la pantalla **Figura 4.21 Certificado no encontrado.**

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. ***La extensión*** ha creado el certificado autentificador.
2. ***La extensión*** no puede guardar el certificado autentificador en storage.
3. ***La extensión*** muestra al usuario el siguiente mensaje "No se pudo guardar el certificado en el Storage" en la pantalla **Figura 4.20 Certificado no guardado en Storage.**

... Fin de la Trayectoria Alternativa 2 ...

Caso de uso: PII_CU4. Generación de Chaffing	
Concepto	Descripción
Actor	-
Propósito	Este caso de uso es el encargado de generar la petición modificada, es decir, de combinar el certificado con la petición HTTP mediante el método de Chaffing que decidimos aplicar.
Entradas	Certificado autentificador y Petición HTTP válida.
Salidas	Petición HTTP modificada.
Pre-condiciones	Haber terminado el caso de uso <b>PII_CU4. Interceptar Petición</b> .
Post-condiciones	-
Reglas del negocio	PII_RN1. Petición válida.
Errores	No se tenga una petición HTTP válida. No se cuente con un código autentificador válido.

Cuadro 4.9: Descripción CU: PII\_CU4

... Trayectoria Principal ...

1. ***La extensión*** Ha recibido una petición HTTP válida.
2. ***La extensión*** Tiene un certificado guardado en storage.
3. ***La extensión*** aplica el algoritmo de Chaffing en el encabezado de la petición HTTP ocultando el certificado en los headers de ésta.
4. ***La extensión*** libera la petición modificada al servidor.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***La extensión*** no ha encontrado el certificado autentificador en storage.

2. ***La extensión*** muestra al usuario el siguiente mensaje "No se encuentra certificado" en la pantalla **Figura 4.21 Certificado no encontrado.**

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. ***La extensión*** no ha recibido ninguna petición HTTP válida.
2. ***La extensión*** no realiza ninguna acción ya que, aunque se cuente con el certificado al carecer de petición HTTP es imposible generar una petición modificada con Chaffing.

... Fin de la Trayectoria Alternativa 2 ...

Caso de uso: PII_CU5. Liberación de Petición.	
Concepto	Descripción
Actor	Navegador de Google Chrome
Propósito	Este caso de uso permite al Navegador de Google Chrome liberar la petición una vez que haya sido modificada por la extensión. Dentro de la petición se incluye el código autenticador junto con el patrón a utilizar en el proceso de <i>Winnowing</i> una vez que sea recibido por el servidor.
Entradas	Petición
Salidas	Petición modificada enviada en red al servidor.
Pre-condiciones	Haber completado exitosamente el proceso de <i>Chaffing</i> al igual que la creación del patrón, e injectarlo en la petición.
Post-condiciones	Con los datos generados (Código Chaffing y patrón) generados por la extensión, se deben de injectar en la petición para poder ser enviada al servidor a través de la red.
Reglas del negocio	PII_RN1 PII_RN2 PII_RN3 PII_RN4
Errores	No se cuenta con acceso a internet Hay un error de estructura en la petición

Cuadro 4.10: Descripción CU: PII\_CU5

### ... Trayectoria Principal ...

1. ***El Navegador Google Chrome*** realiza una petición con el encabezado HTTP modificado, el cual contiene dentro del apartado *headers* del encabezado HTTP el certificado modificado con el proceso de *Chaffing* y el patrón que ayudará al servidor al momento de hacer el proceso de *Winnowing*.
2. ***El analizador Wireshark*** captura la petición generada y muestra el encabezado modificado. Esto con el fin de mostrar que se ha generado el certificado y se ha modificado exitosamente con el proceso *Chaffing*.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *El Navegador de Google Chrome* no puede establecer una conexión con el servidor debido a que no se cuenta con acceso a internet.
2. *El analizador Wireshark* no podrá obtener los datos del encabezado, ya que no capturará la petición.

... Fin de la Trayectoria Alternativa 1 ...

## 2.3. Diagrama de flujo (DF).

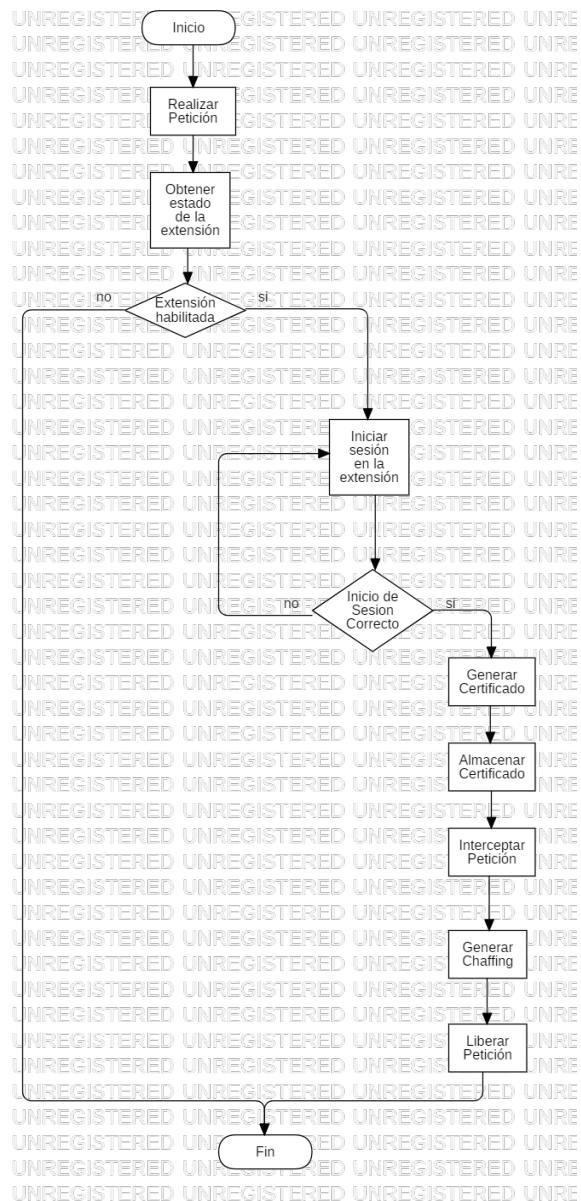


Figura 4.16: Diagrama de flujo del Prototipo 2.

### **2.3.1. Descripción diagrama de flujo.**

Para el caso de este diagrama se inicia con una petición realizada por el navegador web para luego analizar si la extensión se encuentra activada, en caso de que no no se realiza ninguna acción, pero si si se encuentra se pedirá que se inicie sesión en la extensión con el propósito de generar un certificado válido para el usuario, mas tarde será necesario almacenar este certificado en la extensión y se procederá a interceptar la petición HTTP para modificarla, por lo que se genera el chaffing mediante la combinación de la petición y el certificado para luego liberar la petición modificada al servidor.

## 2.4. Diagrama de flujo de datos (DFD).

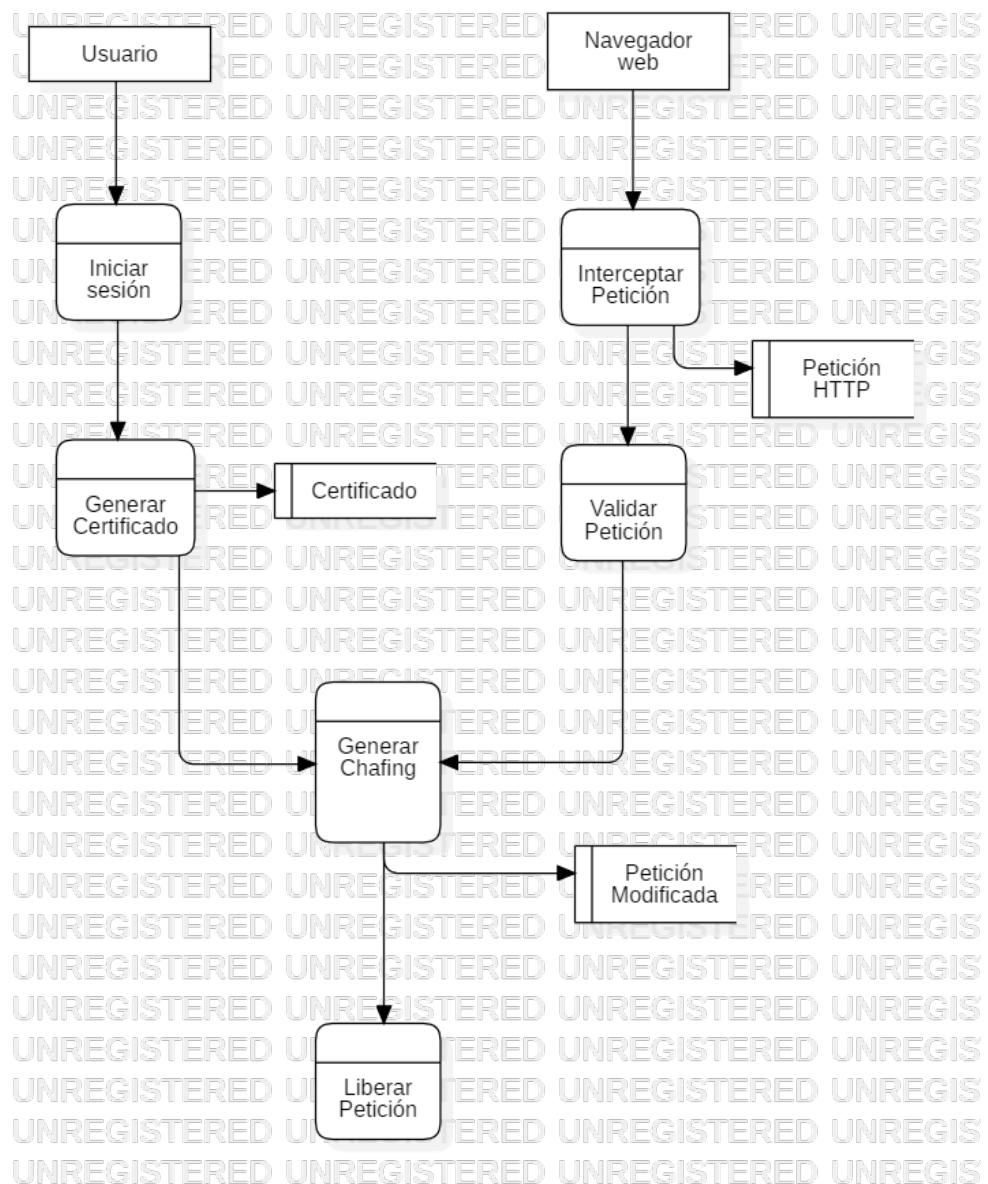


Figura 4.17: Diagrama de flujo de datos de Prototipo 2.

#### **2.4.1. Descripción diagrama de flujo de datos.**

En este caso contamos con dos entidades externas, el usuario por una parte debe iniciar sesión para que se pueda generar un **Certificado** el cual se utilizará para generar el chaffing y por otro lado el Navegador web que intercepta una **petición HTTP** que de igual manera se utilizará para generar el chaffing. Como resultado de la mezcla de estos dos se genera una **petición modificada** la cuál mas tarde se liberará para que pueda viajar hacia el servidor.

## 2.5. Diagrama de clases.

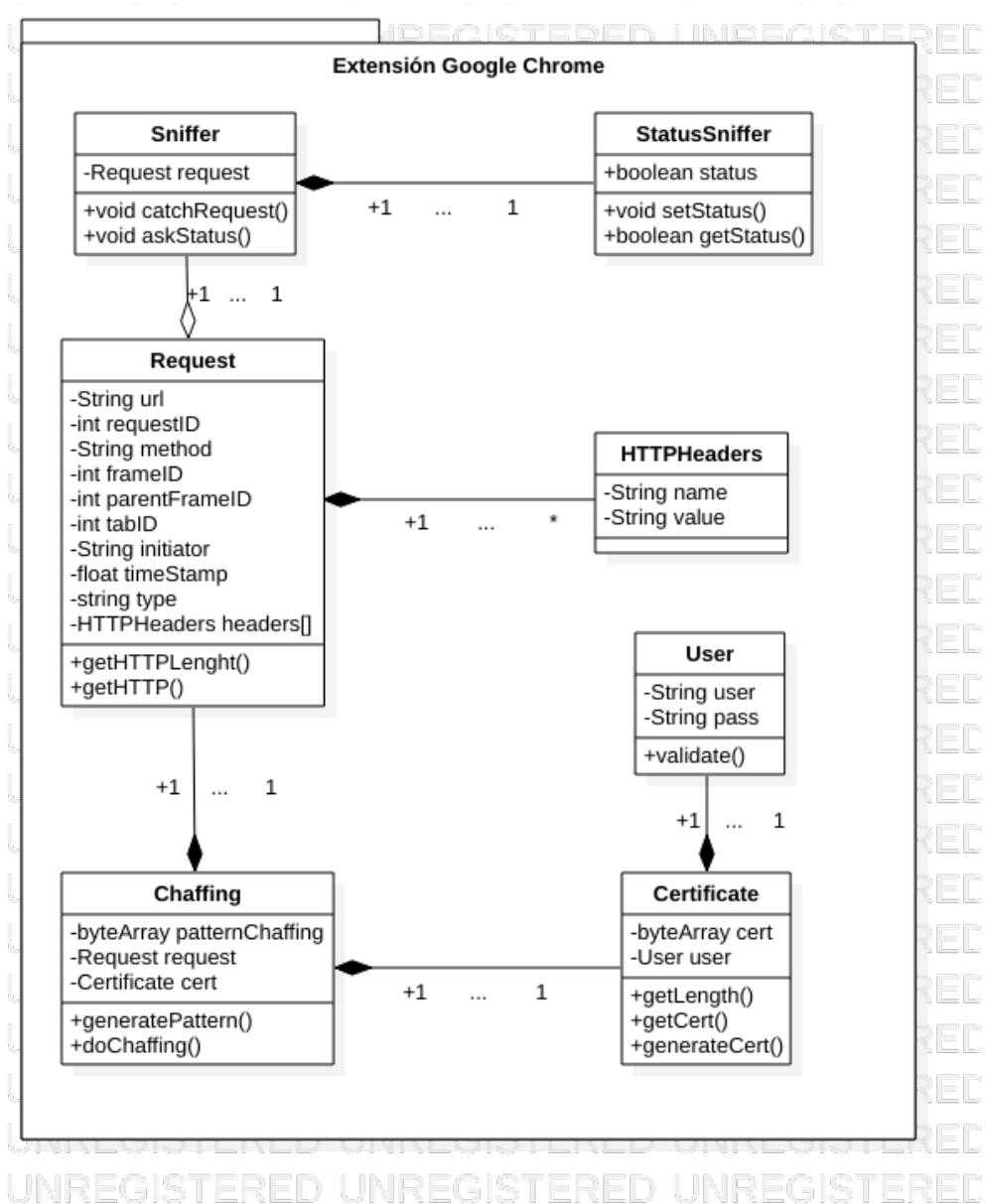


Figura 4.18: Diagrama de clases de Prototipo 2.

### 2.5.1. Descripción de diagrama de clases

User

1. **user** : Variable que permite almacenar el valor del campo nombre de usuario.
  - a) Tipo de dato: **String**.
  - b) Longitud de caracteres: **20 caracteres**.
  - c) Restricciones: La cadena sólo puede tener símbolos alfanuméricos
2. **pass** : Variable que permite almacenar el valor del campo contraseña.
  - a) Tipo de dato: **String**.
  - b) Longitud de caracteres: **16 caracteres**.
  - c) Restricciones: Al menos un **número y una letra mayúscula**. La cadena sólo puede tener símbolos alfanuméricos.

Métodos

- a) **validate()** : Método que valida que los valores de las variables sean válidos.

Certificate

1. **cert** : Variable que permite almacenar el valor del certificado autenticador.
  - a) Tipo de dato: **byte []**.
  - b) Longitud: **32 bytes**.
2. **user** : Variable que permite almacenar el usuario actual que se está utilizando.
  - a) Tipo de dato: **User**.

Métodos:

- a) `getLength()` : método que devuelve la longitud del certificado.
- b) `getCert()` : método que devuelve el certificado.
- c) `generateCert()` : método que genera el certificado con base a los datos del usuario.

Chaffing

- 1. **patternChaffing** : Variable que permite almacenar el patrón de *chaffing*.
  - a) Tipo de dato: `byte []`.
  - b) Longitud: `HTTP.Length() + cert.Length()`.
- 2. **request** : Variable que permite almacenar la petición HTTP.
  - a) Tipo de dato: `Request`.
- 3. **cert** : Variable que permite almacenar el certificado del usuario.
  - a) Tipo de dato: `Certificate`.

Métodos:

- a) `generatePattern()` : método que genera el patrón de *chaffing*.
- b) `doChaffing()` : método que realiza el chaffing en la petición HTTP.

## 2.6. Diagramas de secuencia.

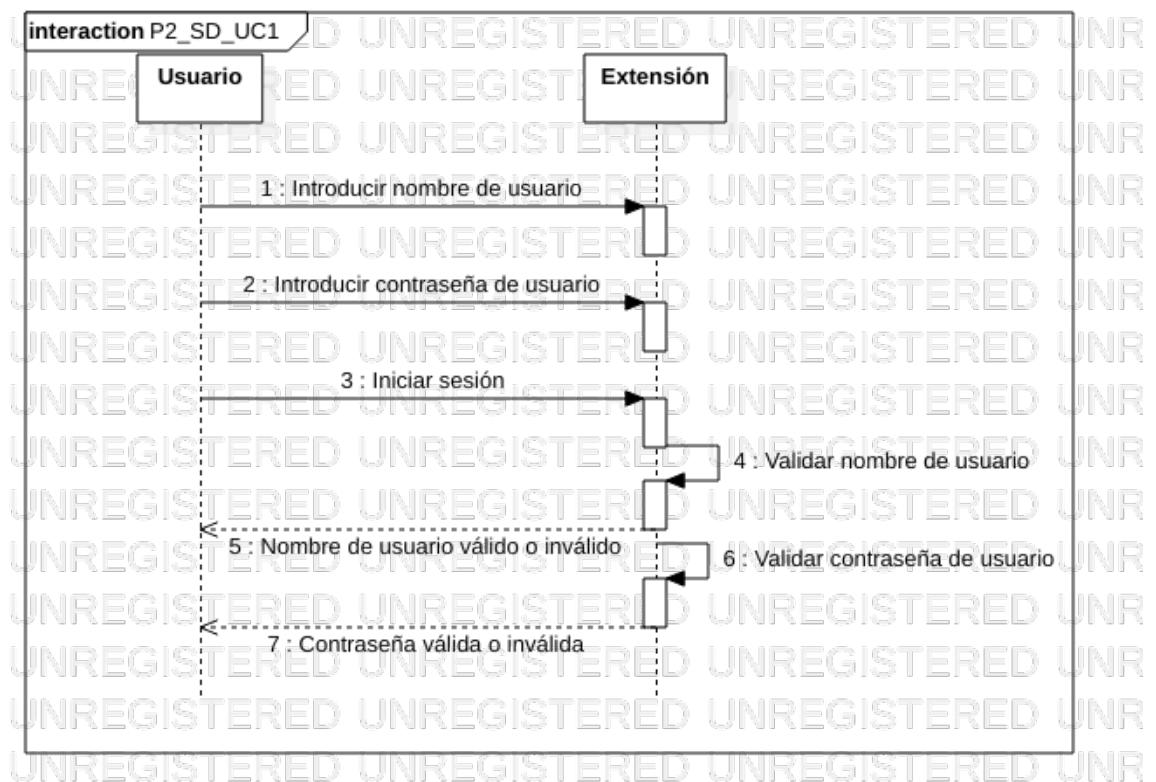


Figura 4.19: Diagrama de secuencia del PII\_CU1. Inicio de sesión en la extensión.

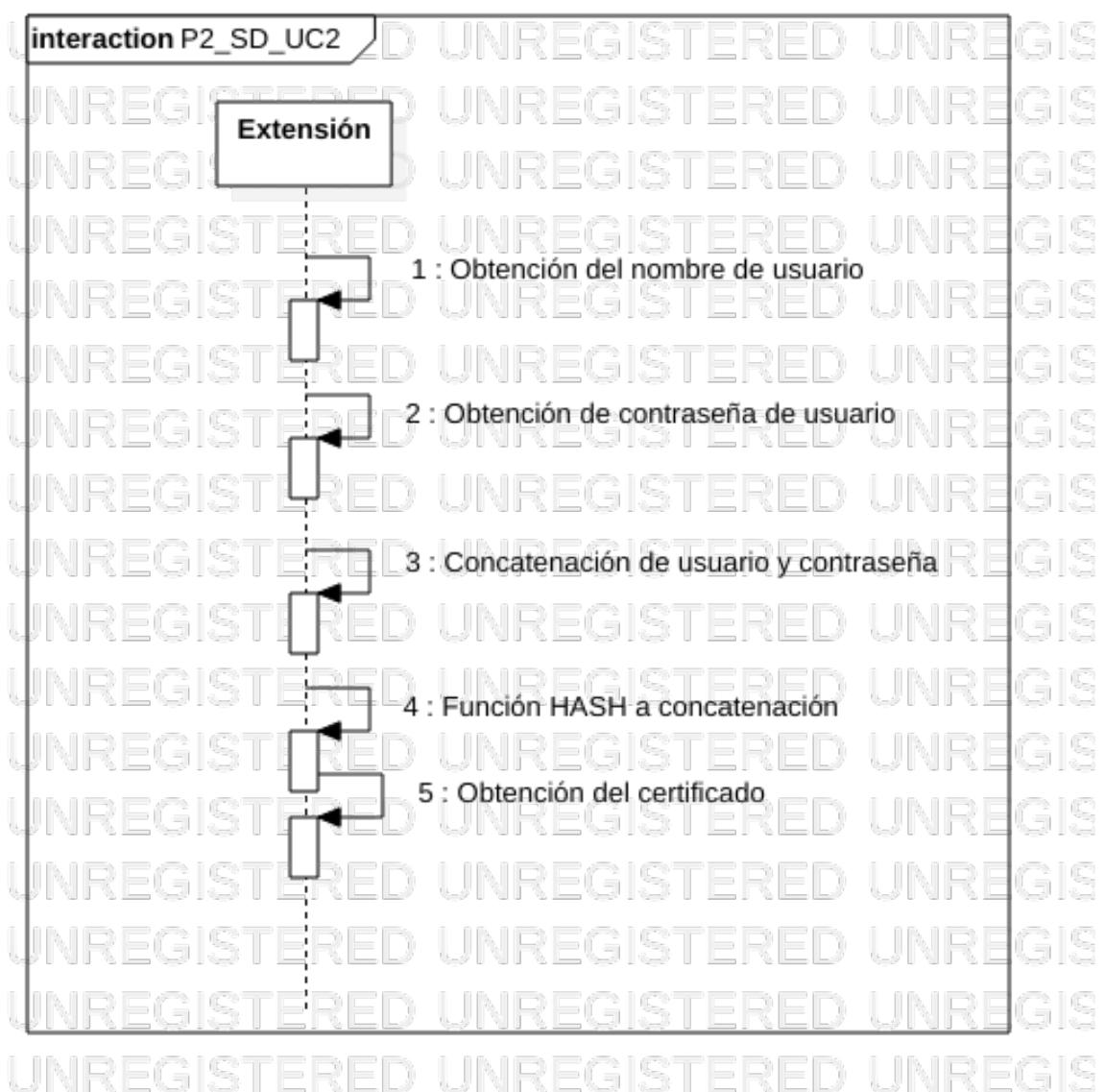


Figura 4.20: Diagrama de secuencia del PII\_CU2. Generación del certificado autentificador.



Figura 4.21: Diagrama de secuencia del PII\_CU3. Almacenamiento del certificado autenticador.

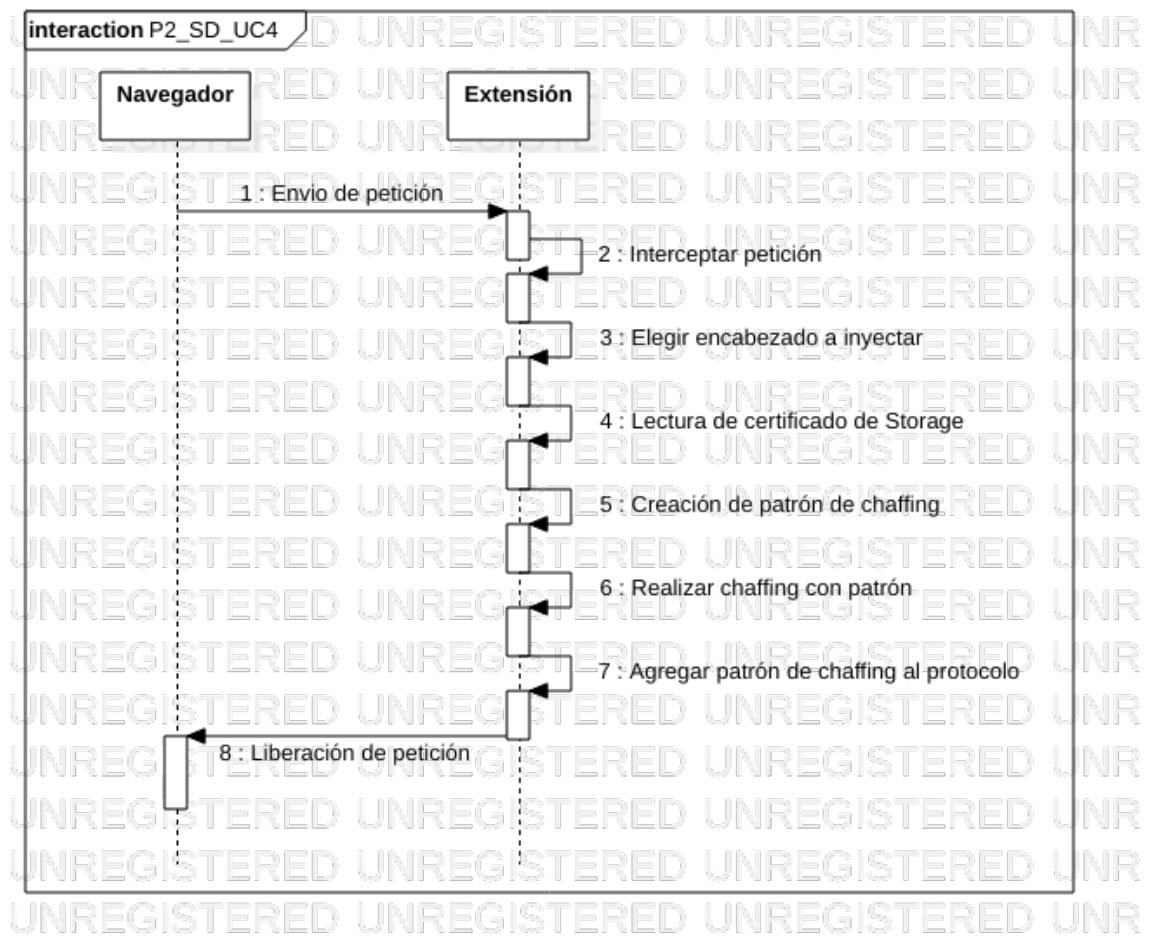


Figura 4.22: Diagrama de secuencia del PII\_CU4. Generación del chaffing.

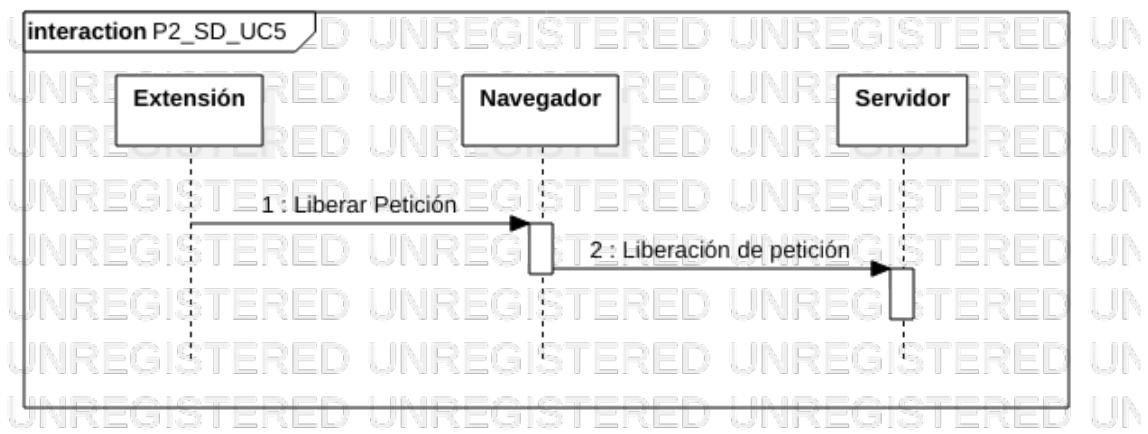


Figura 4.23: Diagrama de secuencia del PII\_CU5. Liberación de la petición.

## 2.7. Diagrama de actividades

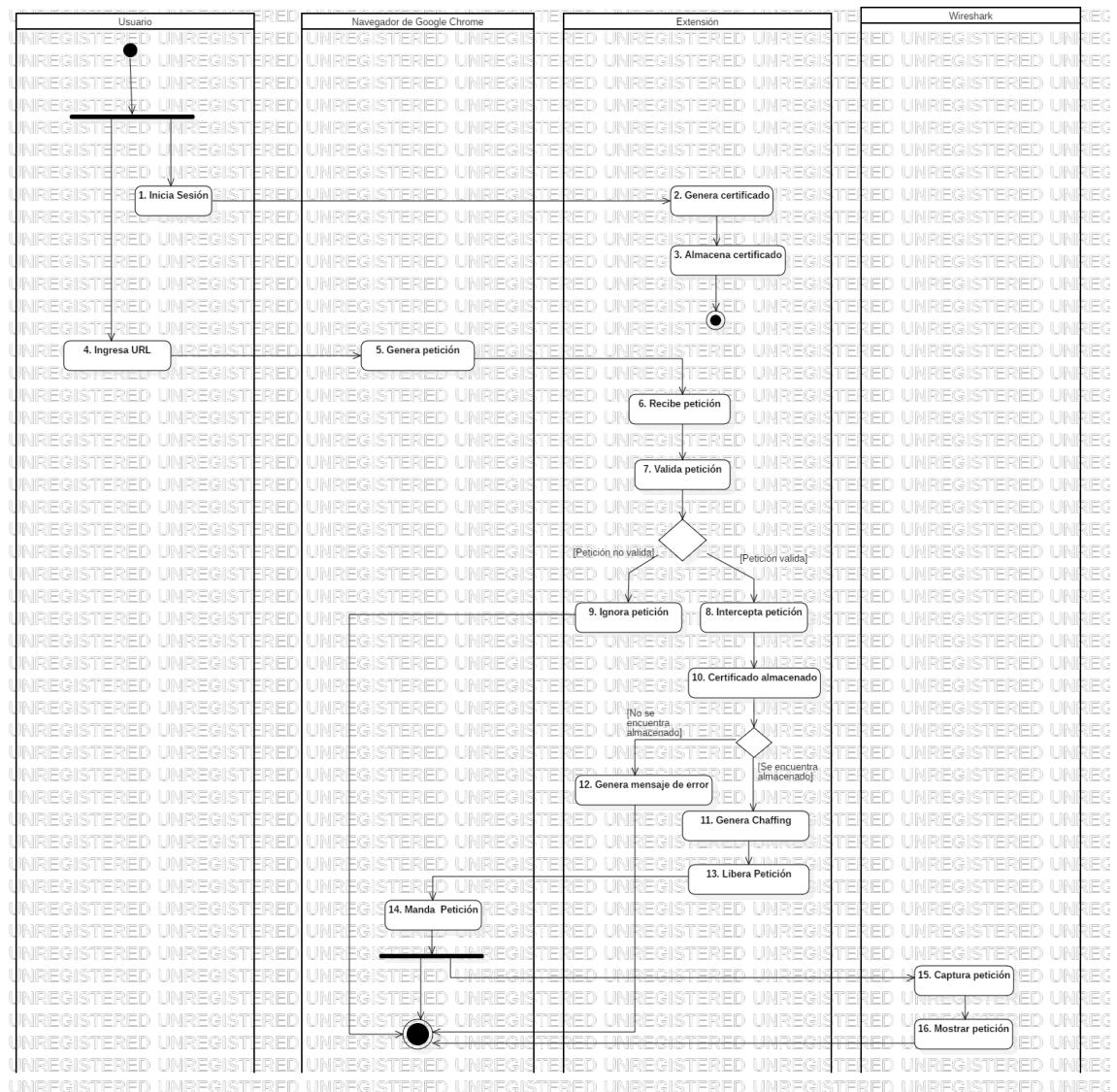


Figura 4.24: Diagrama de actividades del Prototipo 2.

### 2.7.1. Descripción del diagrama de actividades.

Para este prototipo se han agregado más pasos para el diagrama de actividades, los cuales se describen a continuación. Uno de los primeros pasos que debe hacer el usuario es iniciar

sesión, esto con el fin de generar un certificado de acuerdo a los datos del usuario. Dicho certificado se almacena en la API de Google Chrome *Storage*.

El otro camino a seguir por el usuario es realizar una búsqueda en el navegador, en la cual el navegador realiza la petición y la extensión recibe la misma. Modificándola siempre y cuando se encuentre un certificado guardado en *Storage*. Si este proceso es válido, se genera el proceso de Chaffing y se libera la petición. El servidor se encarga de mandar dicha petición almacenada y en donde el analizador *Wireshark* se encarga de capturar la petición para mostrarla.

## 2.8. Interfaz de usuario.



Figura 4.25: Pantalla inicial.

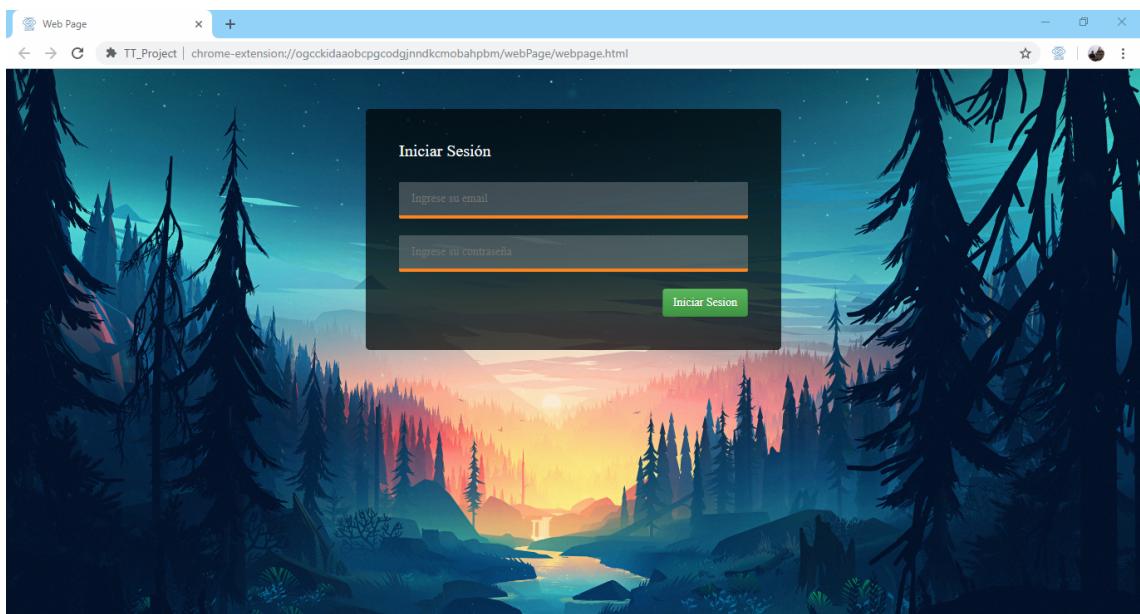


Figura 4.26: Tab de la extensión. Permite inicio de sesión



Figura 4.27: Mensaje de éxito al guardar el certificado en storage de la extensión.

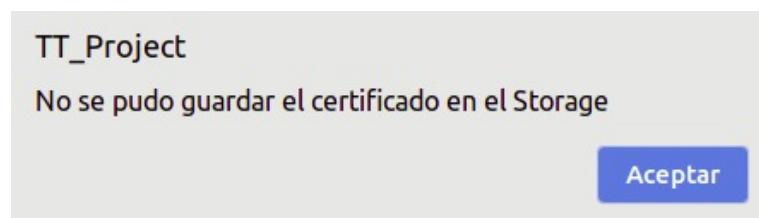


Figura 4.28: Mensaje de error al guardar el certificado en storage de la extensión.

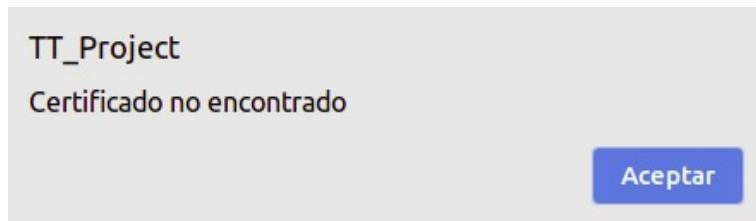


Figura 4.29: Mensaje de error al obtener certificado del storage de la extensión.

## 2.9. Requisitos de diseño.

En este apartado, se especificarán los requisitos de diseño para que el prototipo opere de forma correcta, de igualmanera se tiene por entendido que son necesarios los Requisitos del diseño del primer prototipo.

### 2.9.1. Requisitos de ejecución de la extensión

Para poder ejecutar el prototipo dos es necesario contar con todo lo requerido para el prototipo uno y agregarle la interacción con **jQuery 3.3.1** y **Bootstrap** que son dos poderosos frameworks que corren sobre JavaScript y que nos permiten interactuar un poco mejor con el usuario haciendo la interfaz de la extensión más amigable y entendible, estos ya se encuentran insertados en la extensión por lo que no es necesario que el usuario realice acción alguna. Otros de los requisitos para su ejecución es contar con un **usuario y contraseña** válidos ya que serán de suma importancia para el correcto funcionamiento del prototipo dos.

### 2.9.2. Requisitos para el envío del código autentificador

Para este prototipo se considera necesaria los siguientes requisitos de diseño:

- **Código Autentificador** Archivo indispensable en este prototipo debido a que sin este es imposible acompletar el proceso.
- **Botón para subir archivo** Este botón tendrá el propósito de darle al usuario la opción de escoger su certificado con el cual podrá ingresar a todas sus cuentas.
- **Botón de chaffing** Botón que generará un patrón de chaffing donde se aplicará en la petición HTTP, se inyectará el certificado de forma segura en la petición, y se simulará su envío al servidor.

# Capítulo 5

## Pruebas.

### 1. Prototipo I.

Para verificar el funcionamiento de este primer prototipo, procederemos a habilitar la extensión y realizar distintas peticiones para verificar que la extensión bloquee la salida de la petición y la imprima en otra pestaña del navegador.

Como primera prueba, se desactivará la extensión para mostrar que se puede utilizar el navegador de manera común.

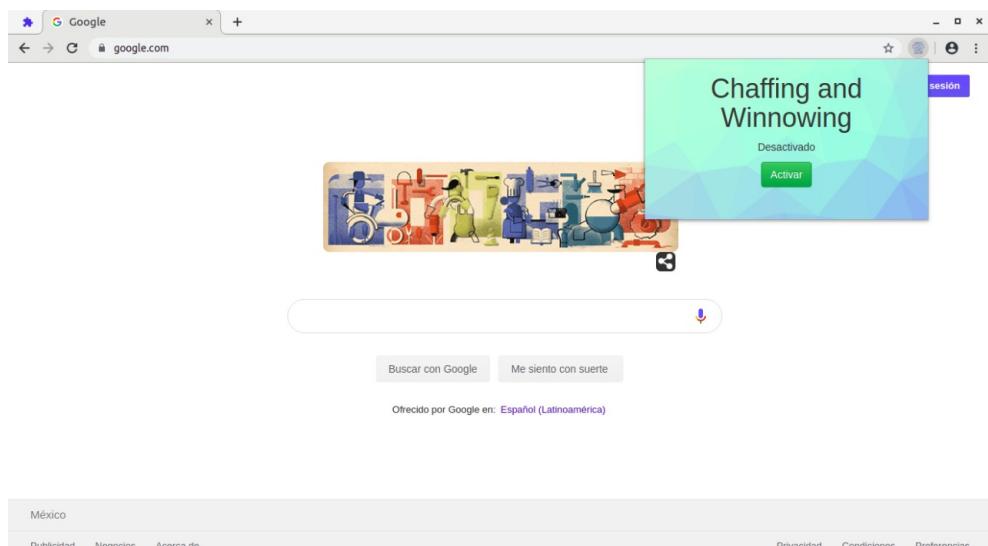


Figura 5.1: Servicio desactivado. Acceso a página [www.google.com](http://www.google.com)

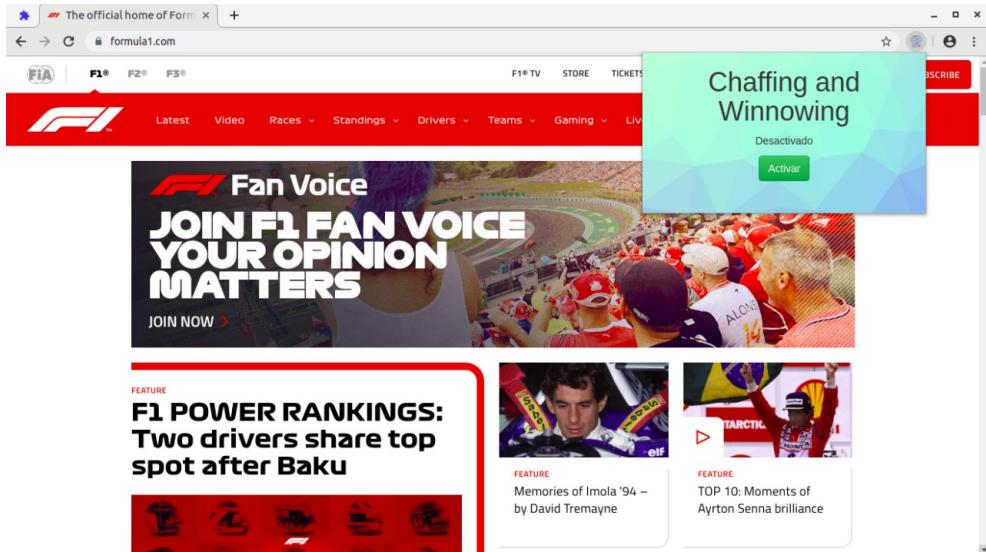


Figura 5.2: Servicio desactivado. Acceso a página www.formula1.com

Una vez activada la extensión, la misma empezará a bloquear las peticiones que se hagan a través del navegador *Google Chrome*. Sea cual sea la petición, la extensión la bloqueará.

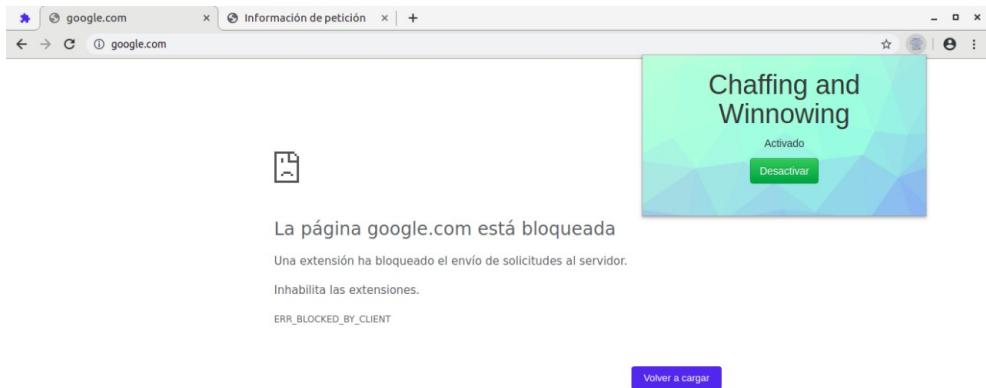
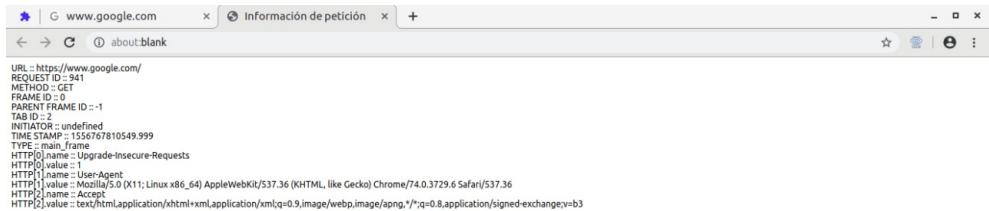


Figura 5.3: Servicio activado. Acceso denegado a www.google.com

La petición bloqueada será mostrada en una nueva pestaña del navegador donde se muestra todo el encabezado. Se trabajara sobre este encabezado la inyección del código autenticador en el Prototipo II.



The screenshot shows a browser window with the title bar "Información de petición". The address bar shows "www.google.com". The main content area displays the following text:

```
URL::https://www.google.com/
REQUEST ID::941
METHOD::GET
FRAME ID::-
PARENT FRAME ID::-
TAB ID::-
INITIATOR::undefined
TIME STAMP::1556767810549.999
TYPE::main_frame
HTTP[0].name::Upgrade-Insecure-Requests
HTTP[0].value::1
HTTP[1].name::User-Agent
HTTP[1].value::Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.6 Safari/537.36
HTTP[2].name::Accept
HTTP[2].value::text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
```

Figura 5.4: Servicio activado. Muestra de petición *www.google.com* intercepcionada en otra pestaña.

La prueba se hace nuevamente, pero ahora con otra página, en este caso se intenta acceder a la página [www.formula1.com](http://www.formula1.com).



Figura 5.5: Servicio activado. Acceso denegado a [www.formula1.com](http://www.formula1.com)

```
URL::https://www.formula1.com/
REQUEST ID::942
METHOD::GET
FRAME ID::-
PARENT FRAME ID::-
TAB ID::-
INITIATOR::undefined
TIME STAMP::1556768080760.742
TYPE::main_frame
HTTP[0].name::Upgrade-Insecure-Requests
HTTP[0].value::1
HTTP[1].name::User-Agent
HTTP[1].value::Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.6 Safari/537.36
HTTP[2].name::Accept
HTTP[2].value::text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
```

Figura 5.6: Servicio activado. Muestra de petición [www.formula1.com](http://www.formula1.com) interceptada en otra pestaña.

## 2. Prototipo II.

Para verificar que nuestro prototipo funciona correctamente procedemos a mostrar una prueba del funcionamiento del mismo.

Primeramente se necesita iniciar sesión en la extensión, esto con el fin de obtener un certificado y que el mismo se guarde en el storage de Google Chrome. En la prueba iniciamos sesión con los siguientes datos:

- **usuario:** usuarioPrueba
- **contraseña:** usuarioP1234@

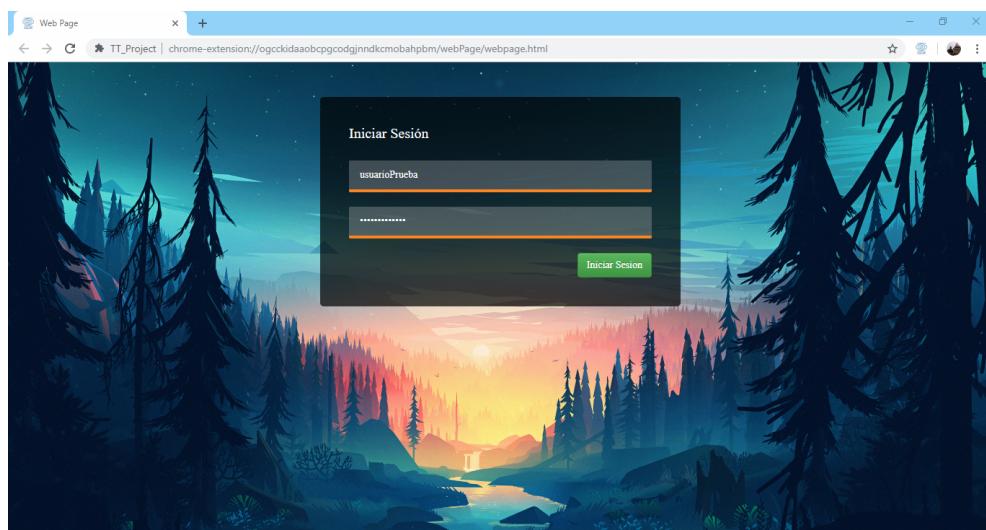


Figura 5.7: Inicio de sesión para la obtención del certificado.

Generando así el siguiente código hash:

**3cc09605f15d36fcd7b14cc7795921fe**

Como siguiente paso, este código hash se guarda en el *Storage* de Google Chrome y se da aviso del estado de este procedimiento.

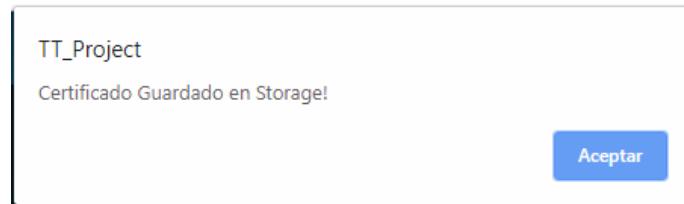


Figura 5.8: Aviso de guardado del certificado en Storage.

Una vez que el certificado se haya guardado con éxito en el *Storage* de Google Chrome, realizamos una búsqueda en Google Chrome, para esta prueba se hizo una petición a un servidor local, montado en *Apache Tomcat* previamente instalado en otra maquina dentro de una red LAN<sup>1</sup>. Esta petición será bloqueada por la extensión para injectar el código con *Chaffing* en el protocolo HTTP en el apartado de *headers*.



### La página 10.0.0.8 está bloqueada

Una extensión ha bloqueado el envío de solicitudes al servidor.

Inhabilita las extensiones.

ERR\_BLOCKED\_BY\_CLIENT

[Volver a cargar](#)

Figura 5.9: Pagina bloqueada del lado del cliente.

---

<sup>1</sup>Una red de área local o LAN (Local Area Network) es una red de computadoras que abarca un área reducida a una casa, un departamento o un edificio.

Una vez injectado el certificado autenticador, se envía la petición con el protocolo modificado al servidor destino. Para verificar que a través de la red se envía la petición modificada se hace uso del analizador de paquetes Wireshark.

La siguiente imagen muestra la petición enviada al servidor, donde en el apartado de *Hypertext Transfer Protocol* se observa que en **Accept** está inyectado el certificado con *Chaffing* y en **Pattern** está inyectado el patrón realizar el *winnowing* en el servidor (en el siguiente prototipo).

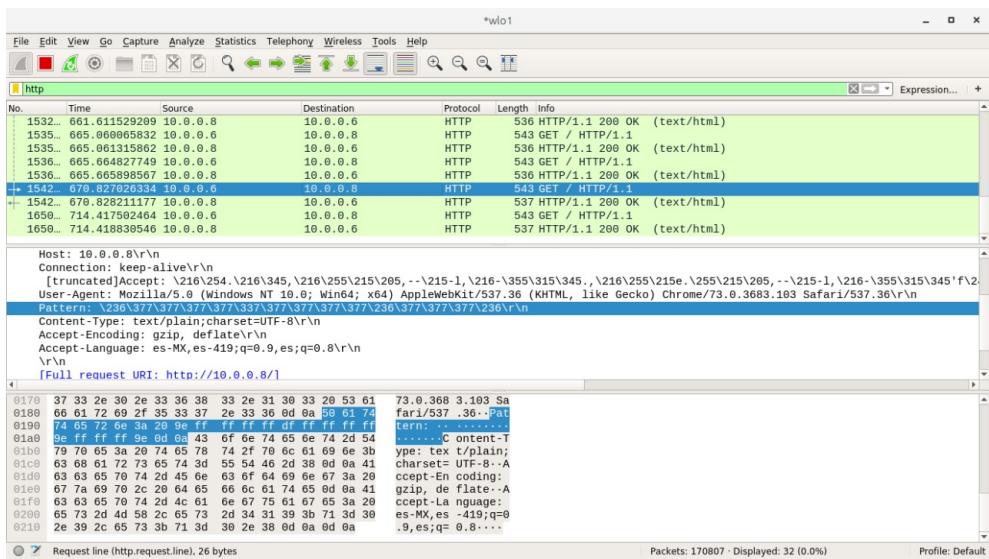


Figura 5.10: Wireshark analizando en red WI-FI filtrando análisis por *http*.

Para un análisis más específico, se muestra la petición en una herramienta que proporciona wireshark, *Show Packet Bytes...*, la cual muestra el encabezado completo, teniendo una herramienta para seleccionar la codificación del texto. Nosotros elegimos la codificación ISO8859-1, ya que en ésta nos permite ver de manera legible el encabezado.

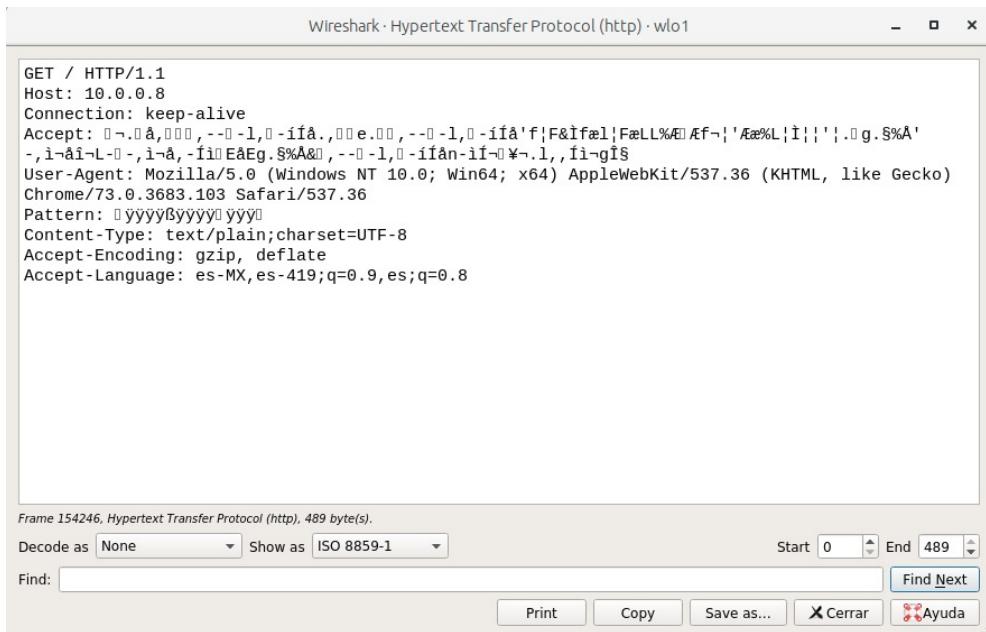


Figura 5.11: Análisis de petición en *Show Packet Bytes...*

# Capítulo 6

## Conclusión

Al terminar este trabajo nos pudimos percatar de todas las nuevas tecnologías que existen y de lo complejo que es ocultar un mensaje no reconocido en las cabeceras HTTP de las peticiones que viajan a través de internet, así como de la complejidad que este Trabajo terminal implica ya que al diseñar la arquitectura completa del sistema analizamos cada uno de los bloques que lo iba a conformar y lo complejo que era cada uno de ellos, pero también aprendimos mucho acerca de cuestiones de seguridad en las extensiones de Google Chrome, por otro lado investigamos mucho sobre mecanismos de autentificación y la manera en la que ellos trabajan sobre la red buscando ser lo más seguro posible para todos los usuarios y así mismo confirmamos la necesidad de un método que nos proporcione confidencialidad como lo es Chaffing and Winnowing.

A lo largo del desarrollo de los dos primeros prototipos tuvimos muchas dificultades sobre como íbamos a enviar la petición HTTP modificada con el código autentificador y su respectivo patrón sin lograr que cualquier usuario pueda identificarlos fácilmente por lo que realizamos varios cambios a la idea principal de nuestro algoritmo y resultó uno mucho más seguro y mas confidencial de lo que esperábamos. Estamos conscientes de que llevamos implementada una pequeña parte de todo el sistema

pero tenemos claro cómo lo vamos a realizar y la complejidad que este representa.

## Capítulo 7

### Trabajo a futuro (TT2)

Como segunda parte de este Trabajo Terminal se realizará el análisis y desarrollo de los prototipos 3 y 4 que representan dos de los bloques más importantes de este trabajo terminal, por el lado del prototipo 3 se implementara un servicio web que implementará una API con el propósito de realizar el Winnowing de la petición HTTP modificada por lo que implica una modificación al servidor Apache para que reconozca este tipo de peticiones, por otro lado se implementará un método de cifrado asimétrico mediante RSA para comunicar el patrón resultante de Chaffing de manera segura entre la extensión y el servicio web, de esta manera buscamos autenticar al usuario de manera rápida, eficiente y sencilla para los usuarios. Por otro lado el prototipo 4 consiste en la implementación de un servidor autenticador que será el encargado de crear y almacenar cada uno de los certificados correspondientes para cada usuario con el fin de que éstos los puedan obtener fácilmente en su extensión logrando que se puedan iniciar múltiples sesiones en diferentes extensiones, siempre cuidando que este certificado se encuentre lo más seguro posible. Como podemos analizar todavía nos falta una gran parte de este extenso trabajo pero confiamos en nuestro equipo y en nuestra habilidad para poder resolver todas las problemáticas que se nos presenten en el camino.

# Bibliografía

- [1] A. Komarova y A. Menshchikov, Comparison of Authentication Methods on Web Resources, St. Petersburg, Russia: St. Petersburg National Research University of Information, 2016.
- [2] A. Beutelspacher, Cryptology, Washington, D.C.: Mathematical Association of America, 1994.
- [3] J. R. Aguirre, Seguridad Informática y Criptografía, Madrid, España: Universidad Politécnica de Madrid, 2006.
- [4] A. J. Menezes, P. v. Oorschot y S. Vanstone, Handbook of Applied Cryptography, FL, USA: CRC Press, 1996.
- [5] S. D. Santiago, Generacion De Sucesiones Criptograficamente Fuertes, México, Df: Uam , 2005.
- [6] A. Maiorano, Criptografia: Tecnicas De Desarrollo Para Profesionales, Buenos Aires, Argentina: Alfaomega, 2009.
- [7] Mozilla,«JavaScript» 22 Abril 2018. [En línea]. Disponible: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> [Último acceso: 15 Enero 2019].
- [8] J. D. Gauchat, El gran libro de HTML5, CSS3 y Javascript, Barcelona: MARCOMBO, S.A., 2012.

- [9] UNAM, «Aspectos Básicos de la Seguridad en Aplicaciones Web,» 19 Abril 2016. [En línea]. Disponible: <https://www.seguridad.unam.mx/historico/documento/index.html-id=17>. [Último acceso: 15 Enero 2019].
- [10] UNAM, «Robo de Identidad y Consecuencias Sociales,» 16 Junio 2011. [En línea]. Available: <https://www.seguridad.unam.mx/historico/documento/index.html-id=16?fbclid=IwAR0u8WAXORvBxZ3H-aMzlBhd-6o7g8ycS88eRu7nY1t1XVtCufhEcQ7hWDs>. [Último acceso: 2019 Febrero 15].
- [11] Aguilar, A., Hernández, A. 25 de Abril de 2014). Obtenido de Sugerencias de Seguridad para Sitios Web: <http://www.seguridad.unam.mx/documento-id=1143>
- [12] Fundación Wikimedia, Inc., «Cookie (informática),» 23 abril 2019. [En línea]. Available: [https://es.wikipedia.org/wiki/Cookie\\_\(informatica\)](https://es.wikipedia.org/wiki/Cookie_(informatica)). [Último acceso: 2019 abril 28].
- [13] allaboutcookies.org, «¿Para qué se utilizan las cookies permanentes?,» 2019. [En línea]. Available: <http://www.allaboutcookies.org/es/galletas/cookies-persistentes-utilizados-para.html>. [Último acceso: 2019 abril 28].
- [14] S. Juliá, «Qué son las cookies, tipos de cookies y cómo cumplir la ley,» 2019. [En línea]. Available: <http://www.gadae.com/blog/que-son-las-cookies-tipos-de-cookies-y-como-cumplir-la-ley/>. [Último acceso: 28 abril 2019].
- [15] Oficina de Seguridad del Internauta, «Entre cookies y privacidad,» 18 julio 2018. [En línea]. Available:

<https://www.osi.es/es/actualidad/blog/2018/07/18/entre-cookies-y-privacidad>. [Último acceso: 28 abril 2019].

- [16] Universidad Politécnica de Madrid, «Criptografía,» 2004. [En línea]. Available: [http://www.dma.fi.upm.es/recursos/aplicaciones/matematica/\\_discreta/web/aritmetica/\\_modular/criptografia.html](http://www.dma.fi.upm.es/recursos/aplicaciones/matematica/_discreta/web/aritmetica/_modular/criptografia.html). [Último acceso: 20 abril 2019].
- [17] University of Bath, «Dissertation,» 2007. [En línea]. Available: <http://www.cs.bath.ac.uk/ mdv/courses/CM30082/projects.bho/2007-8/durongdej-r-dissertation-2007-8.pdf>. [Último acceso: 28 abril 2019].
- [18] The Pennsylvania State University, «Rivest Seguridad,» 2019. [En línea]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.4853&rep=rep1&type=pdf>. [Último acceso: 28 abril 2019].
- [19] S. GOLDWASSER AND S. MICALI, “Probabilistic encryption,” Journal of Computer and System Science, Vol. 28, 1984, pp. 270–299.
- [20] R. L. Rivest, «Chaffing and Winnowing: Confidentiality without Encryption,» 18 marzo 1998 . [En línea]. Available: <https://people.csail.mit.edu/rivest/Chaffing.txt>. [Último acceso: 15 febrero 2018].
- [21] Springer International Publishing AG 2018 A. Abraham et al. (eds.), Proceedings of the Second International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’17), Advances in Intelligent Systems and Computing 679, DOI 10.1007/978-3-319-68321-8\_11

- [22] Mozilla. (2019, 19 marzo). FileSystem. Recuperado 29 abril, 2019, de <https://developer.mozilla.org/es/docs/Web/API/FileSystem>
- [23] VeriSign, Inc., «Todo lo que debe saber sobre certificados SSL,» 2019. [En línea]. Available: [https://www.verisign.com/es\\_LA/website-presence/online/ssl-certificates/index.xhtml](https://www.verisign.com/es_LA/website-presence/online/ssl-certificates/index.xhtml). [Último acceso: 28 abril 2019].
- [24] GlobalSign, «¿Qué es SSL?,» 2019. [En línea]. Available: <https://www.globalsign.com/es/centro-de-informacion-ssl/que-es-ssl/>. [Último acceso: 20 abril 2019].
- [25] Google code, «crypto-js,» 2019. [En línea]. Available: <https://code.google.com/archive/p/crypto-js/#MD5>. [Último acceso: 20 abril 2019].
- [26] wireshark, «What is Wireshark?,» 2019. [En línea]. Available: [https://www.wireshark.org/docs/wsug\\_html\\_chunked/ChapterIntroduction.html#ChIntroWhatIs](https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs). [Último acceso: 20 abril 2019].
- [27] O. S. OCAÑA, ANÁLISIS, ESTUDIO Y DESARROLLO DE CRIPTOGRAFÍA DE CURVAS ELÍPTICAS, Mexico,DF: UNAM, 2008. Recuperado de: [http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/10047/Tesis\\_Completa.pdf?sequence=1](http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/10047/Tesis_Completa.pdf?sequence=1)
- [28] Administración de IBM TRIRIGA Application Platform «Autenticación de usuarios mediante el inicio de sesión único,» 2019. [En línea]. Available: [https://www.ibm.com/support/knowledgecenter/es/SSHEB3\\_3.6.0/com.ibm.tap.doc/sso\\_topics/t\\_ctr\\_authenticate.html](https://www.ibm.com/support/knowledgecenter/es/SSHEB3_3.6.0/com.ibm.tap.doc/sso_topics/t_ctr_authenticate.html). [Último acceso: 03 mayo 2019].

- [29] IBM Community «TRIRIGA Wiki Home,» 2019. [En línea]. Available: <https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20TRIRIGA1>. [Último acceso: 03 mayo 2019].