Método de autentificación mediante Chaffing and Winnowing en el protocolo HTTP

Bryan Israel Blancas Pérez, Gerardo Carrillo Fernández, Diego Arturo Morales González, Pedro Antonio Paredes Hernández, Axel Ernesto Moreno Cervantes, Sandra Díaz Santiago,
Escuela Superior de Cómputo I.P.N. CDMX
Tel. 5729-6000 ext. 52000 y 52021. E-mail: bapb97@gmail.com, gerardocf.ipn@gmail.com, diegoarturo2121@gmail.com, live_13@live.com.mx

***Resumen*—En este trabajo terminal se busca verificar la viabilidad de implementar la técnica Chaffing and Winnowing como una alternativa de autentificación en una aplicación web con inicio de sesión por contraseña haciendo uso de certificados digitales expedidos por una autoridad certificadora en un servidor autentificador, una extensión de Google Chrome™del lado del usuario y una API del lado del servicio web para lograr que los usuarios pueda autentificarse de forma automática y segura.**

***Palabras Clave*—Método de Autentificación, Chaffing and Winnowing, Extensiones de Google Chrome™**

I. Introducción

Hasta el 2017 la autentificación por contraseña ha sido la más utilizada en los servicios web por su facilidad de implementación, mantenimiento y usabilidad para el usuario. Sin embargo, mientras esta autentificacion nos brinda la posibilidad de implementarla fácilmente, no es tan segura como otros métodos y requiere la memorización de las contraseñas [1]. Como consecuencia de ello, los servicios web han implementado mecanismos de seguridad tales como contraseñas con un mínimo de caracteres, al menos un carácter especial, entre otros, lo que ha provocado el surgimiento de herramientas de administración de contraseñas o directamente al almacenado de éstas en medios físicos ó digitales que podrían ser inseguros [2].

I-0a. Protocolo HTTP: El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML). entre un navegador y un servidor web localizado mediante una cadena de caracteres denominada dirección URL [3].

I-0b. Certifiado: Un certificado digital (X.509, v.3) es un fichero que contiene la clave pública y privada de una persona y está avalado (firmado electrónicamente) por una entidad de confianza o *Autoridad de Certificación*. Todo certificado debe estar firmado electrónicamente por una Autoridad de Certificación (AC) para ser valido [4].

I-0c. Métodos de autentificación: Para poder asegurar la confidencialidad de la información manejada en los servicios web, 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 [1].
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.[5]

A continuación en el cuadro 1, mostraremos algunas comparativas para tener una mejor perspectiva de las ventajas, desventajas, vulnerabilidades de los diferentes métodos de autenticación:

	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

Figura 1. Tabla comparativa de los diferentes métodos de autentificación.

I-0d. 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 estenografía [6].
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 [7].

II. Metodología

La metodología utilizada para llevar a cabo el proceso de desarrollo es la metodología por ”**Prototipos Evolutivos**” con pequeñas adaptaciones que se realizaron para el mejor funcionamiento en el proceso de realización del sistema. Esta metodología ayuda a los desarrolladores a mejorar la

comprensión del trabajo a elaborar cuando los requerimientos no están completamente definidos o pueden ir cambiando a lo largo de la implementación . Así mismo, el paradigma brinda la posibilidad de utilizar fragmentos de programas existentes o aplicar herramientas que permitan generar rápidamente proyectos que funcionen y puedan evolucionar [8].
Al utilizar este paradigma de desarrollo se busca que, mediante la implementación parcial del trabajo, surgan requerimientos no contemplados desde el planteamiento del problema, 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.
Para este trabajo terminal se desarrollaron 2 prototipos, debido a que el prototipo 2 es más rápido en los procesos de *Chaffing and Winnowing* que el prototipo 1. Teniendo como prototipo final a el prototipo 2. A continuación se expica el desarrollo de ambos prototipos.

PROTOTIPO 1.

La arquitectura general del sistema para el prototipo 1 se muestra en la figura 2, donde se puede observar como es la interacción entre los componentes.

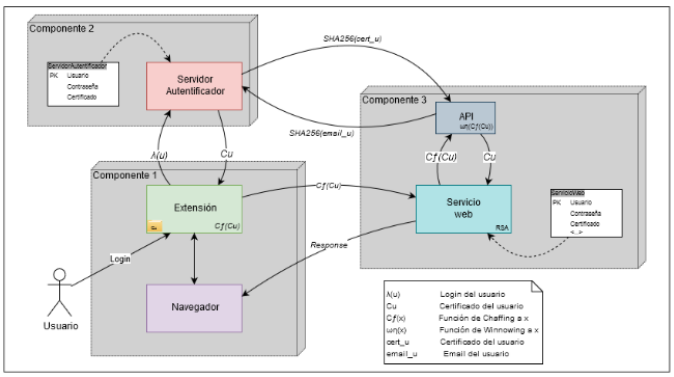

Figura 2. Arquitectura general del sistema del prototipo 1.

El sistema se compone de 3 grandes bloques los cuales se comunicarán vía red:

1. **Navegador Chrome con la Extensión instalada**: Este componente permite a la extensión poder interceptar peticiones hechas por el usuario a través del navegador de Google Chrome. Una vez que se intercepte la petición, ésta podrá ser modificada y liberada. La modificación se hará sólo mientras la extensión esté habilitada, y tiene como objetivo inyectar el certificado autentificador en el encabezado del protocolo una vez que se haya llevado a cabo el proceso de Chaffing. Cuando dicho certificado sea inyectado, la extensión libera la petición para que salga a red.
   Este certificado será único por cada usuario y puede ser obtenido desde el Componente 2 (Autoridad certificadora) cuando el usuario inicie sesión en la extensión. Si el usuario no puede iniciar sesión debido a que no tiene una cuenta, desde la extensión el usuario tiene la opción de registrarse para poder crear su cuenta y a su vez su certificado.
   Además, el usuario podrá cerrar la sesión en la extensión si es que así lo desea, lo que elimna el certificado de la máquina local del usuario. Por otro lado, si el usuario desea revocar su certificado, lo puede hacer también desde este componente.
   Para inyectar el certificado autentificador, se crea un *"patrón de chaffing"*, este patrón se genera aleatoriamente para después mandarlo junto con la petición HTTP. Dicho patrón viaja cifrado con la clave pública del Componente III (API). El objetivo de mandar el patrón junto con el protocolo HTTP, es que el servidor sea capaz de descifrar el patrón con su clave privada y con él realizar la etapa de *Winnowing* para extraer el certificado.
   El algoritmo para la generación del patrón de *Chaffing* se muestra en la figura 3 el cual es el encargado de generar pseudo-aleatoriamente una cadena de 1's con una lognitud de 150 bytes, este patrón necesita una cota mínima y una cota máxima para saber cuantos bytes válidos (1's) serán colocados en el patrón.

Data: lenCert, lenAccept, Cert
Result: patternChaffing
$lenPattern \longleftarrow lenCert + lenAccept;$
$Pattern[lenPattern * 8] \longleftarrow 0;$
$unosCert \longleftarrow getOnes(Cert);$
for $i = 1$ **to** $unosCert$ **do**
  repeat
    $random \longleftarrow secure\_random(0, lenPattern);$
  until $Pattern[random]! = 0;$
  $Pattern[random] \longleftarrow' 1';$
end

Figura 3. Generación de patrón de Chaffing.

El algoritmo utilizado para la generación de chaffing se muestra en la figura 4. En ella podemos observar que se recorre el patrón un número de veces definido para completar el chaffing. Al momento de hacer el recorrido sobre el patrón se verifica si se trata de un byte válido (1) para colocar un byte del certificado o un byte no valido (0) para colocar un byte chaff.

2. **Servidor autentificador:** Este componente es el encargado de crear y almacenar los certificados de los usuarios. En este servidor, los usuarios tienen registrada una cuenta y a la cual pueden obtener desde el Componente 1 (Extensión) su certificado. Una vez que el usuario inicie sesión en la extensión, este servidor autentificador regresará como respuesta el certificado generado para que la extensión lo almacene. Para la creación del certificado autentificador se utiliza la herramienta OpenSSL, además, la comunicación entre extensión y servidor se hace bajo SSL/TLS.
   La principal función de este componente es gestionar las cuentas y certificados, almacenando el correo electronico, contraseña y certificado del usuario en su base de datos, de tal forma que el Componente 1 se pueda comunicar con este componente para la generación de un certificado, mandar su certificado

```
Data: Cert, Accept
Result: Chaffing
lenCert ⟵ lenght(Cert);
lenAccept ⟵ lenght(Accept);
Pattern ⟵ getPattern(lenCert, lenAccept, Cert);
Chaffing ⟵ "";
countCert ⟵ 0;
countAccept ⟵ 0;
certBits ⟵ getBits(Cert);
acceptBits ⟵ getBits(Accept);
foreach i in Pattern do
    if i ==' 1' then
        Chaffing ⟵ Chaffing + certBits[countCert];
        countCert ⟵ countCert + 1;
    else
        Chaffing ⟵ Chaffing + acceptBits[countAccept];
        countAccept ⟵ countAccept + 1;
    end
end
Chaffing ⟵ getBytes(Chaffing);
```

Figura 4. Generación de Chaffing por medio del patrón.

si es que ya se encuentra registrado en el servidor, o bien, para revocar un certificado. Así mismo, este componente también estará comunicado con el Componente 3, específicamente la API, para controlar la revocación de certificados.
El propósito de este componente es poder crear y utilizar un certificado real, generado por una autoridad certificadora de confianza y con el cual se pueda autentificar a un usuario en un servicio web.

3. **Servidor web con API instalada:** Para el componente 3, se utiliza un servicio web de prueba para mostrar la funcionalidad de inicio de sesión por este método propuesto y donde se realiza la etapa de *Winnowing* a la petición HTTP para obtener el certificado. Esto último lo realiza una API dedicada exclusivamente a ello, con conexión al Componente 2 para checar el estatus y validez del certificado. El servicio web sólo debe conectarse al API para obtener el certificado y su validez, con base en ello realizar la lógica del negocio para iniciar sesión. Así mismo, el API implementa un cifrado asimétrico (RSA), esto con la finalidad de que el Componente 1 pueda tomar la llave pública del servidor y cifrar el *"patrón de chaffing"* que se manda en la petición. Una vez que el patrón y el chaffing llegue al API, sólo ella puede descifrar el patrón con su llave privada, dandole la integridad necesaria al patrón para viajar a través de la red.
El propósito de este componente es poder obtener el certificado que identificará a cada usuario, para poder validarlo y saber si se debe de dar o negar el acceso. Cabe mencionar que, la primera vez que el servidor reciba el certificado autentificador en el servicio web, éste pedirá al usuario que se inicie sesión con la finalidad de poder asociar este certificado a una cuenta del servicio; para las peticiones posteriores, el certificado ya contará con una cuenta asociada a la cual podrá dar acceso siempre y cuando el certificado sea válido.

El algortimo que se desarrollo para el proceso de Winnowing se muestra en la figura 5 el cual se implementa en la API. Como sabemos el proceso de *Winnowing* es el proceso análogo al de *Chaffing*, donde se necesita conocer el patrón para saber que bytes son válidos y cuales no lo son. Obteniendo del *Chaffing* sólo los bytes válidos, al término de este proceso se obtendrá el certificado en texto plano, para así comprobar junto con la Autoridad certificadora si este certificado que llegó a traves de la petición es válido o no.

```
Data: chaffing, patternCipher, aesCipher, privateKey
Result: cert,header
chaffingDecode[] ⟵ base64.decode(chaffing);
aesKey ⟵ rsa.decipher(aesCipher, privateKey);
pattern[] ⟵ aes.decipher(patternCipher, aesKey);
cert[];
header[];
i ⟵ 0;
while i < pattern.length do
    if pattern[i] == 1 then
        header.add(chaffingDecode[i]);
    else
        cert.add(chaffingDecode[i]);
    end
    i ⟵ i + 1;
end
```

Figura 5. Obtención de certificado a traves del proceso de Winnowing.

PROTOTIPO 2.

La arquitectura general del sistema para el prototipo 2 tuvo unas pequeñas modificaciones como se muestra en la figura 6, donde se puede observar como es la interacción entre los componentes.

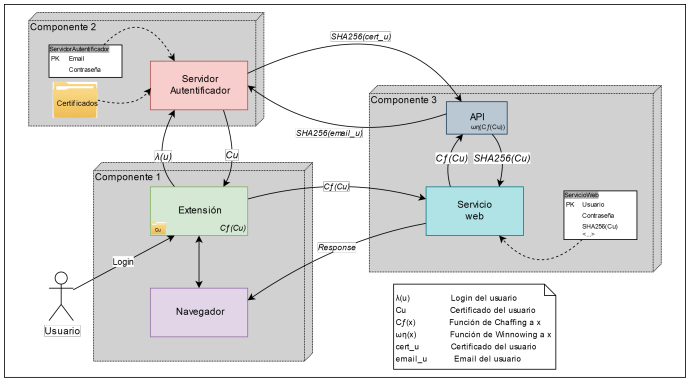

Figura 6. Arquitectura general del sistema del prototipo 2.

Como vemos el sistema se compone de los mismos componentes que en el prototipo 1 pero con ciertas modificaciones las cuales se mencionan a continuación:

1. **Extensión de Google Chrome™:** Para el caso de la extensión se optó por *eliminar el cifrado AES* con lo que ahora el patrón deberá reducir su tamaño para ser cifrado por RSA sin complicaciones, por lo que se *cambió el algoritmo de generación del patrón* como se muestra en

la figura 7 y al mismo tiempo hubo modificaciones al *algoritmo de generación del código chaffing* como se muestra en la figura 8

2. **Servidor autentificador:** En este caso se agregó un *servidor FTP* con control de acceso para agregar un filtro más de seguridad en la obtención de los mismos.
3. **API:** Al modificar el patrón se tuvieron consecuencias en la API empezando por la *eliminación del cifrado AES* y la modificación del *algoritmo de winnowing* como se muestra en la figura 9 así mismo, la API ahora retorna un hash del certificado al servicio web con lo que podrá autentificar al usuario sin la necesidad de tener la responsabilidad de almacenar el certificado real por lo que ahora el único responsable de almacenar los certificados de los usuarios es el *servidor autentificador*.

```
Data: low, high
Result: pattern,ones
lenPattern ⟵ 150 ∗ 8;
ones ⟵ rand(low, high);
pattern[lenPattern] ⟵ 0;
i ⟵ 0;
while i < ones do
    x ⟵ securerand(size);
    if pattern[x]! = 1 then
        pattern[x] ⟵ 1;
        i ⟵ i + 1;
    end
end
```

Figura 7. Generación de patrón de Chaffing del Prototipo 2.

```
Data: Cert
Result: Chaffing,pattern
lenCert ⟵ lenght(Cert);
pattern, ones ⟵ getPattern(550, 650);
Chaffing ⟵ "";
rep ⟵ roof(lenCert/ones);
i ⟵ 0;
k ⟵ 0;
while i < rep do
    foreach j in pattern do
        if j ==' 1' then
            Chaffing ⟵ Chaffing + Cert[k];
            k ⟵ k + 1;
            ones ⟵ ones − 1;
        else
            Chaffing ⟵ Chaffing + fake();
        end
        if ones == 0 then
            break;
        end
    end
    i ⟵ i + 1;
end
```

Figura 8. Generación de Chaffing por medio del patrón del prototipo 2.

III. Resultados

Se analizó el funcionamiento de cada uno de los prototipos mencionados en la sección anterior para determinar la viabilidad del método de autentificación, al ser un sistema de tres componentes como se muestra en la figura 2, fue necesario realizar *pruebas de integración* en donde principalmente se presentan los datos enviados y *pruebas de funcionamiento* de cada uno de los componentes en donde se comprueba el correcto funcionamiento de cada uno. así como los tiempos de los algoritmos involucrados.

```
Data: chaffing, patternCipher, privateKey, sizeCert
Result: cert,header
pattern[] ⟵ rsa.decipher(patternCipher, privateKey);
cert ⟵ "";
rep ⟵ sizeCert/numOnes(pattern);
contRep ⟵ 0;
while i < rep do
    while j < pattern.length do
        if cert.length == sizeCert then
            break;
        end
        if patt[j] == 1 then
            cert+ = chaffing[i + (pattern.length ∗ contRep)];
        end
        j ⟵ j + 1;
    end
    i ⟵ i + 1;
end
```

Figura 9. Obtención de certificado a traves del proceso de Winnowing del prototipo 2.

III-0a. Pruebas de integración: Para el caso de las pruebas de integración se analizó la interacción entre cada uno de los elementos de los componente como se presenta a continuación:

- **Extensión y Servidor Autentificador:** Esta interacción se realiza en el inicio de sesión cuando el usuario obtiene el certificado para almacenarlo del lado del cliente a lo que todo funcionó con éxito, en el caso de usar conexión SSL un capturador de tramas no podrá ver nada de la información que viaja en la red ya que se encuentra cifrada pero de igualmanera los resultados son exitosos como se muestra en la figura 10.
- **Extensión y Servicio Web:** Esta comunicación es de las más importantes ya que es en donde se encuentra el envío del código *chaffing* y el apartado *pattern* para los dos prototipos estos apartados sufrieron cambios pero la prueba continuo siendo exitosa, para comprobarlo se analizó la trama capturada que se muestra en la figura 11.
- **Servidor Autentificador y API:** La principal tarea de esta comunicación es comprobar si los certificados han sido revocados, al no contar con datos sensibles esta comunicación se realiza por HTTP por lo que el analizador de paquetes puede mostrar fácilmente lo que viaja en red que solamente corresponden a el hash del e-mail del usuario y del certificado respectivamente como se muestra en la figura 12.

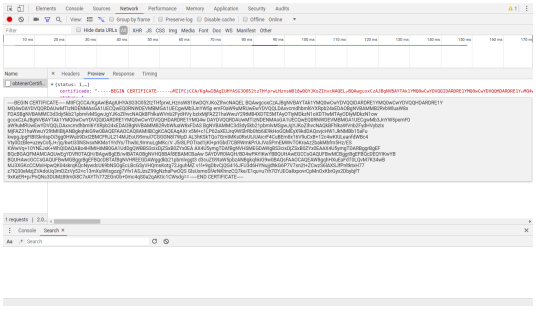

Figura 10. Respuesta de la Autoridad Certificadora hacia la Extensión

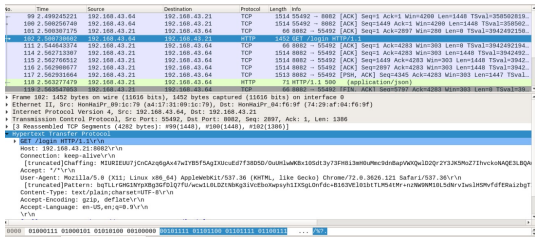

Figura 11. Analizador de protocolos Wireshark de la petición dada por el usuario hacia servicio web.)

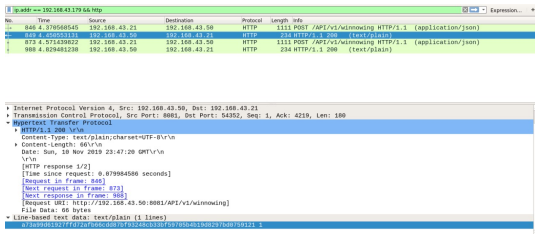

Figura 12. Analizador de protocolos Wireshark de la petición dada por la API hacia la autoridad certificadora.)

III-0b. Pruebas de funcionamiento: Por otro lado se realizaron pruebas sobre cada uno de los componentes para comprobar un correcto funcionamiento de estos de manera independiente a lo que los resultados fueron positivos. Así mismo se realizó una prueba de tiempos para los dos algoritmos principalmente involucrados y uno al inicio de sesión en general y los resultados fueron los siguientes:

- **Algorimo de chaffing:** Para el prototipo 1 que comprende la generación del patrón, generación del código chaffing, cifrado AES y cifrado RSA tomó un tiempo de ejecución de *434.29 ms* mientras que para el prototipo 2 que solo comprendió la generación del patrón, generación del código chaffing y cifrado RSA tomó un tiempo de ejecución de *17.43 ms* con lo que tenemos una diferencia de *416.86 ms*.
- **Algoritmo de winnowing:** Para el prototipo 1 que comprende el decifrado AES, desifrado RSA y el proceso de winnowing se tomó un tiempo de ejecución de *53.99 ms* mientras que para el prototipo dos de este algoritmo, solamente se comprendió el decifrado RSA y el proceso de winnowing con lo que el tiempo de ejecución se redujo a *12.46 ms* con lo que tenemos una diferencia de *41.53 ms*.
- **Inicio de sesión:** Para terminar se realizaron pruebas sobre el inicio de sesión, debido a la latencia de la red se consideró promediar el tiempo de 100 inicios de sesión por lo que los resultados para el prototipo 1 fueron *1101.68 ms* mientras que para el prototipo 2 fue de *335.28 ms* mostrando una diferencia de tiempo de *766.4 ms*

Es importante resaltar que todas las pruebas de tiempo se realizaron en igualdad de condiciones para contar con un resultado más preciso.

IV. Conclusiones

A lo largo del desarrollo de este trabajo, uno de los puntos que se concluye es que no se puede evitar completamente el uso de autentificación por usuario y contraseña, pero en este trabajo se logra proporcionar una alternativa al inicio de sesión de los servicios web que deseen implementar este método de autentificación.

Otro punto a cocncluir, es la recomendación de, aquel servicio web que desee implementar este método debe contar con un certificado SSL para no dejar vulnerable al sistema. Además, debido a las modificaciones que se deben de realizar en la lógica de inicio de sesión del servicio web, no es posible para este trabajo implementarlo en uno que se encuentre en producción, por lo que el desarrollo de este trabajo fue en un servicio web de prueba. Gracias a que la API nos ayuda a conectar softwares, ésta nos proporciona un medio para la comunicación entre servicio web y autoriudad autentificadora, y así mismo llevar a cabo el proceso de Winnowing.

Finalmente, gracias a la metodología implementada, desarrollamos un prototipo funcional que cumplía con nuestro objetivo, además de mejorarlo en cuestión de tiempo de ejecución de los algoritmos, creando así el prototipo final. Uno de los puntos más importantes de esta mejora fue la reducción del tamaño del patrón que se manda hacia el servicio web, cambiando incluso el cifrando que se implementa.

V. Reconocimientos

Los autores de este artículo agradecen a la Escuela Superior de Cómputo del Instituto Politécnico Nacional por el apoyo otorgado para el desarrollo del presente trabajo terminal, en especial a los profesores que participaron como directores y sinodales del mismo por el tiempo, atención y disposición para buscar de la mejor manera que este trabajo contribuyera a nuestra formación. Queremos agradecer a nuestros compañeros que nos apoyaron a lo largo de este trabajo con retroalimentación dada en nuestras exposiciones. Así mismo, querer agradecer a nuestras familias que nos apoyaron para seguir adelante no sólo en este trabajo, si no seguir adelante a lo largo de esta trayectoria académica.

Referencias

[1] A. Komarova y A. Menshchikov, ”Comparison of Authentication Methods on Web Resources” en Proceedings of the Second International Scientific Conference ”Intelligent Information Technologies for Industry”, Varna, Bulgaria, 14–16 Septiembre, 2017, pp 106-120.

[2] C. C. Espinosa Madrigal (2011, junio 16), Robo de Identidad y Consecuencias Sociales, [En línea]. Disponible: https://www.seguridad.unam.mx/historico/documento/index.html-id=16?fbclid=IwAR0u8WAXORvBxZ3H-aMzlBhd-6o7g8ycS88eRu7nY1t1XVtCufhEcQ7hWDs. [Último acceso: 15 de febrero del 2019].

[3] CCM (2017), El protocolo HTTP, [En línea]. Disponible: https://es.ccm.net/contents/264-el-protocolo-http. [Último acceso: 6 Mayo 2019].

[4] L. Hernández Encinas (2015), Las Firmas y los Certificados electrónicos (de la Administración Pública del Estado-CSIC) [En línea]. Disponible: http://www.itefi.csic.es/sites/default/files/hernandez-encinas/firma-y-certificado-electronico.pdf [Último acceso: 6 noviembre 2019]

[5] A. Romero Mier y Terán y M. A. Vasquéz Martínez (2016, abril 19), Aspectos Básicos de la Seguridad en Aplicaciones Web. [En línea]. Disponble: https://www.seguridad.unam.mx/historico/documento/index.html-id=17. [Último acceso: 5 de mayo del 2019].

[6] A. Maiorano, *Criptografia: Tecnicas De Desarrollo Para Profesionales*, 1ra edición, D.F., México: Alfaomega, 2009.
[7] R. L. Rivest, "Chaffing and Winnowing: Confidentiality without Encryption", Laboratorio de ciencias de la computación del MIT, Massachusetts, Estados Unidos de América, 1998.
[8] R. S. Pressman, *Ingeniería del Software. Un enfoque práctico*, 7ma edición, D.F, México: McGraw-Hill, 2010.