Ingeniería Civil en Informática y Telecomunicaciones

Criptografía y seguridad en redes

Tarea 3 Criptografía y seguridad en redes

Autor:
Diego Lagos

Profesor:
Nicolás Boettcherh

26 de Mayo de 2021

Índice

1. Introducción

En el ramo Çriptografía y Seguridad en Redes"de la carrera Ingeniería Civil en Informática y Telecomunicaciones Universidad Diego Portales se les pide a los alumnos hacer variadas tareas.

Este informe comprende la tarea 3 de dicho ramo, donde se le pide a los alumnos aplicar la materia de cifrado simétrico mediante la creación de dos códigos, un Python que genere un script HTML con un mensaje encriptado, y un JavaScript que encuentre el mensaje encriptado, luego lo decripte.

En este informe se narrará el proceso de crear ambos códigos y luego explicarlos de forma en que una persona con conocimientos técnicos básicos pueda comprender.

2. Algoritmos y Tecnologías

Lo primero que se hace al comenzar la tarea es elegir el algoritmo de cifrado y su extensión.

Luego de un poco de investigación se elige el algoritmo AES(Advanced Encryption Standard), sucesor del algoritmo DES. Y "hermano" del algoritmo rijndael.
AES es un sistema de encriptación simétrico por bloque, donde se utiliza el algoritmo original de rijndael junto con cifrado por bloque para crear información encriptada.

Para la extensión, se buscaron algunas comunes pero muchas estaban tomadas por compañeros, entonces se eligió la extensión EAX luego de experimentar con python. EAX es un esquema que utiliza una llave y un nonce envés de un vector inicial (nonce es un número aleatorio que se puede utilizar tanto para encriptar como para desencriptar).

Para poder encriptar mediante AES-EAX se eligió la librería de python llama PyCryptodome, ya que puede generar encriptaciones de EAX sin problema.

Para la libreria de JavaScript se buscó una librería que pudiera encriptar y desencriptat AES-EAX y se encontró en un github una extensión de CryptoJS que hace eso: https://github.com/artjomb/cryptojs-extension

3. Python

Al empezar en python, lo primero que se aprende es generar HTML a través de python mediante las funciones open(archivo, w), write(), close().

```
20
21  f = open('./index.html','w')
22  pagina = """<!DOCTYPE html>
23  <html lang="en">
24  <head>
```

```
67  f.write(s)
68  f.close()
69
```

Luego, se aprende a cifrar un mensaje mediante la librería PyCryptodome.
Para comenzar la encriptación se necesitar 3 valores, el mensaje que se va a encriptar, una llave y un nonce. Estos últimos dos de 16 bits.
En el caso de la llave, ésta se procesó para crear un objeto AES en modo EAX.
Luego, el mensaje claro se encripta mediante la función "encrypt-and-dygest" del objeto, como el resultado estaba en bits, se pasó a base 64 mediante el b64enconde de la librería base64.

```python
from Crypto.Cipher import AES
from string import Template
from base64 import b64encode
from Crypto.Random import get_random_bytes
import json
import base64
import binascii
import webbrowser
import os

llave = b'sixteen byte key'
objeto = AES.new(llave, AES.MODE_EAX)

mensaje = b'sale su lolcito'
textocifrado, tag = objeto.encrypt_and_digest(mensaje)
cifradob64 = '"'+base64.b64encode(textocifrado).decode('utf-8')+ '"'

nonce = b'5684517890481604'
```

Se abre el HTML en modo escribir, y se escribe la página en la variable ”pagina”, luego se emplea la función safe-substitute de la librería string, para reemplazar los valores en el HTML, se escribe el cambio y se cierra la conexión.

```python
s = Template(pagina).safe_substitute(sustituto=cifradob64, algorithm = g, sustituto2 = llave.decode('UTF-8'), sustituto3 = nonce.decode('UTF-8

f.write(s)
f.close()
```

4. HTML

En el HTML se tienen varias líneas de estilo en la parte de ¡header¿, luego se tiene una referencia al script de CryptoJS.

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Tarea 3</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}
.header {
  padding: 80px;
  text-align: center;
  background: #00bfff;
  color: white;
}
.header h1 {
  font-size: 40px;
}
</style>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js" integrity="sha512-nOQuvD9nKirvxDdvQ9
</head>
```

Luego en el ¡body¿se tiene la parte importante, tres líneas visibles para el usuario, y tres líneas no visibles, con los datos que se obtuvieron en el python en la parte "id", con una palabra descriptiva en la parte "class".

```html
<body>
<div class="header">
  <h1>Tarea 3 criptografia</h1>
  <p>Este sitio contiene un mensaje secreto</p>
  <p>Puedes encontrarlo?</p>
  <div class= "AES" id= "AvO9DbsigYUz8eRRKUtS"></div>
  <div class= "llave" id= "sixteen byte key"></div>
  <div class= "nonce" id= 5684517890481604></div>
</div>

</body>
```

5. JavaScript

En la primera parte del JavaScript se referencian los scripts del Github de "cryptojs-extension", el cual agrega el modo AES-EAX a cryptojs.

```
// @require     https://raw.githubusercontent.com/artjomb/cryptojs-extension/master/lib/cryptojs-aes.min.js
// @require     https://raw.githubusercontent.com/artjomb/cryptojs-extension/master/lib/cryptojs-mode-ctr.min.js
// @require     https://raw.githubusercontent.com/artjomb/cryptojs-extension/master/build/eax.min.js
```

Luego se encuentran los valores en la página, empezando por la llave la cual se transforma a hexadecimal, el nonce el cual se transforma a hexadecimal y el mensaje encriptado.

```
16      'use strict';
17      var llave = document.getElementsByClassName("llave");
18      var hexllave = CryptoJS.enc.Hex.parse(llave[0].id);
19
20      var nonce = document.getElementsByClassName("nonce");
21      var hexnonce = CryptoJS.enc.Hex.parse(nonce[0].id);
22
23      var mensaje = document.getElementsByClassName("AES");
24      var mensajestring = mensaje[0].id
25
```

Finalmente se crea un objeto EAX con la llave, para luego usar la función "decrypt" del objeto para desencriptar el mensaje con ayuda del nonce.

```
28      var eax = CryptoJS.EAX.create(hexllave);
29      var textoplano = eax.decrypt(mensajestring, hexnonce);
30
31      console.log(textoplano);
32
```

6. Github

El github conteniendo los códigos está en:
https://github.com/lagossully/CriptografiaTarea3

Mientras que la página html se puede revisar en:
https://lagossully.github.io/CriptografiaTarea3/
En esta última página se puede aplicar el tampermonkey.

7. Conclusión

En esta tarea se aprende como se implementan las encriptaciones simétricas, donde la misma llave sirve para el emisor y el receptor.

A diferencia de la criptografía asimétrica, los tiempos de cálculo de encriptación son cortos, especialmente a grandes volúmenes. En esta tarea ocurrió que al correr el script de Python, la información estaba encriptada en menos de 1 segundo.

Pero la principal desventaja de este tipo de encriptación es el hecho de que la llave da control total de utilizar una encriptación. O sea que el mayor peligro es que una llave caiga en manos equivocadas.

Ya que el cifrado simétrico es rápido y seguro, pero tiene el riesgo mejorado anteriormente, a veces se utilizan otros métodos en forma de apoyo para guardar la seguridad de una llave sensible (ejemplos: hash, cifrado asimétrico).