Resumen Semana 3

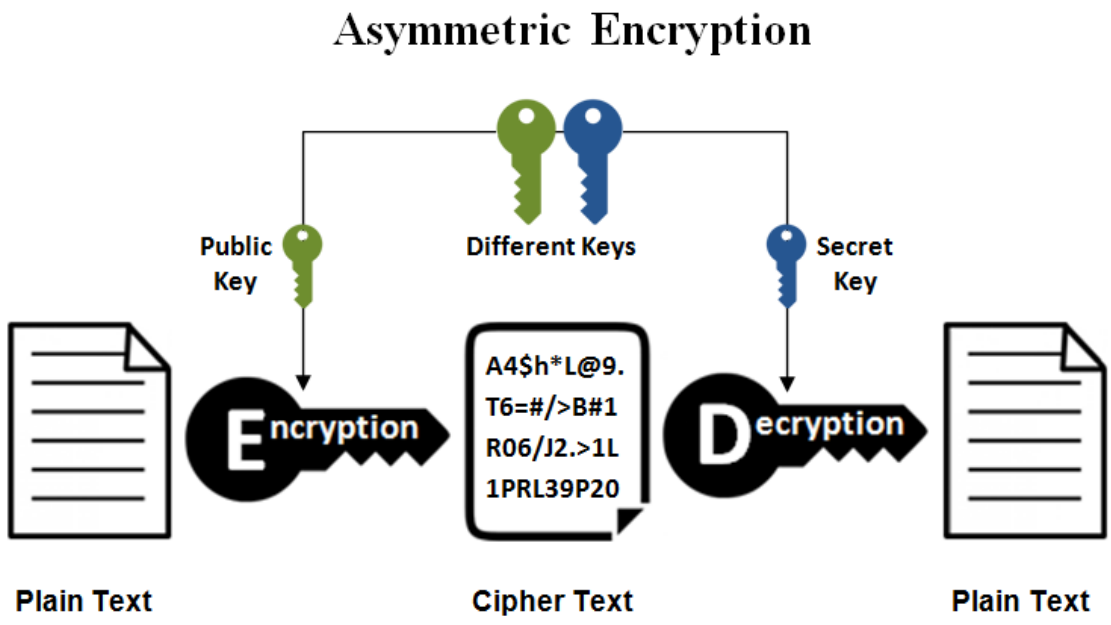
El reto de esta semana consiste en conseguir la flag que habrá en el archivo comprimido que se enviará por Discord y enviársela al bot

Esta semana recordamos lo que son los **cifrados simétrico**:

- El cifrado simétrico es un tipo de cifrado que utiliza una sola clave privada tanto para cifrar como para descifrar.

Y explicamos lo que es un **cifrado asimétrico**:

- Cada usuario posee 2 claves: una pública (la conoce todo el mundo) y una privada (solo la conoce el dueño). Se suele realizar un "Intercambio de Clave", que permite enviar un mensaje cifrado con la **clave pública** del destinatario, y que solo éste, con su **clave privada** pueda descifrarlo.
- Este cifrado se usa extendidamente hoy en día, para establecer conexiones seguras, asegurar la integridad de nuestros mensajes…



Por otra parte, dependiendo de la agrupación de los datos, los cifrados pueden ser de dos tipos:

Cifrado de bloque:

- Se agrupa el mensaje en bloques del mismo tamaño, y se aplican los cifrados sobre dichos bloques. Ex: AES.

Cifrado de flujo:

- Se divide el mensaje en su unidad más pequeña (bits, bytes, caracteres...), y se aplica el cifrados sobre cada unidad. Ex: RC4.

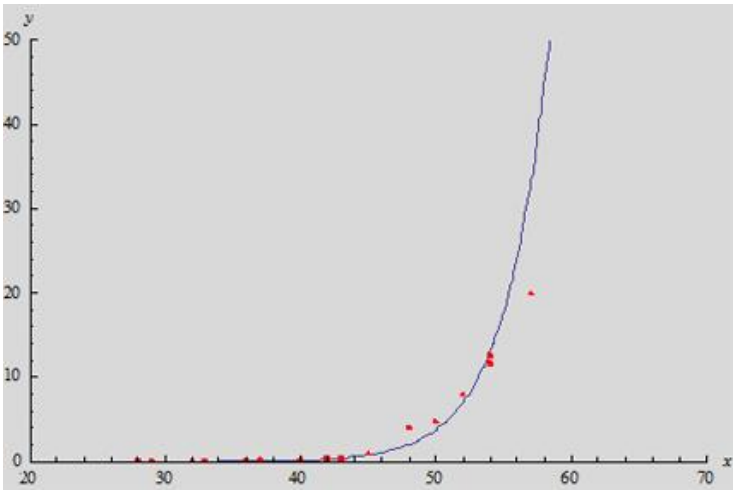
Conociendo esto, vamos a ver:

- El sistema RSA (de manera teórica, porque es bueno conocerlo)
- Hashes (MD5 y SHA-256)
- OSINT

RSA

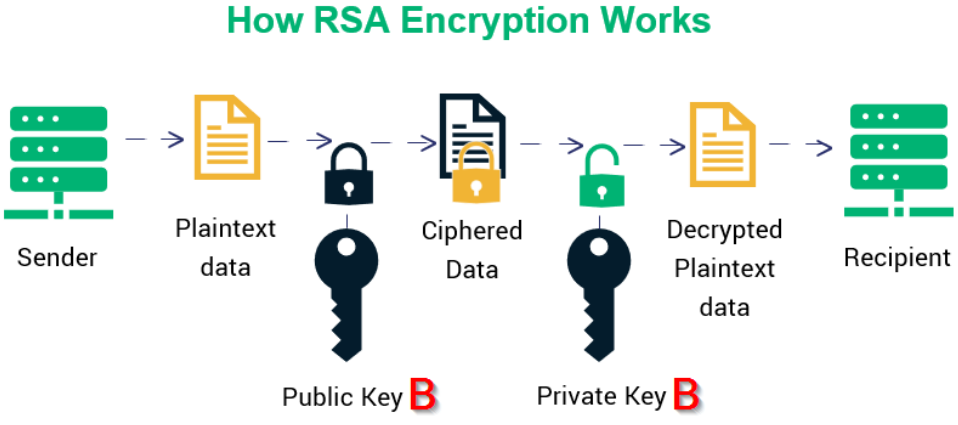
RSA (Rivest, Shamir y Adleman) es un sistema de cifrado asimétrico (cada usuario tiene una clave pública y una privada). La seguridad de RSA se basa en el problema de **factorización de números enteros**.

Para generar sus claves, un usuario primero escoge dos números primos MUY grandes (más de 1024 bits) y hace su producto. Para que un atacante pueda encontrar esa clave privada, tendrá que conocer ambos números primos, y, por ende, factorizar. A partir de cierto número muy grande, intentar factorizar se vuelve un problema exponencial para cualquier ordenador.



La idea matemática detrás de tener dos claves, una pública y una privada, es que estas dos sean inversas.

Entonces, un usuario A puede cifrar un mensaje con la clave pública del usuario B, y sólo el usuario B podrá descifrar el mensaje con su propia clave privada (ya que son inversas).



Hashes

Los hashes son “cifrados” unidireccionales. Esto significa que son **IRREVERSIBLES**. Realmente no son cifrados porque no se pueden descifrar. Un hash es un conjunto de operaciones matemáticas y lógicas (de tipo AND por ejemplo), que al aplicarse a un mensaje lo transforman, pero no se puede volver al mensaje original “descifrando” o usando el mismo algoritmo en sentido inverso.

Vamos a ver algún ejemplo y por qué se usa.

Si usamos la función de hash MD5 (message-digest algorithm) para *hashear* la palabra “maravilloso”, el resultado es:

2ea33c2393199bdceb5e67e49dd47583

Pero, si ahora hacemos lo mismo con la palabra “maravillosa”, a lo mejor esperamos que salga algo parecido, pero sale lo siguiente:

db784747cd90055b64d392227249248d

Una propiedad de los hashes es que, al más mínimo cambio, al menos la mitad de la salida va a cambiar completamente.

Pero, si un hash no se puede descifrar, ¿para qué sirve?

Imagina que tenemos una página web en la que la gente nos compra cosas muy legales. Para que un usuario pueda pagar necesita una cuenta. Y para crearse una cuenta, normalmente pedimos una contraseña (que guardaremos en nuestra base de datos). Como a lo mejor has intuido, guardar contraseñas en texto en claro en una base de datos no es una buena idea. Algo que podemos hacer es guardar el HASH de la contraseña en la base de datos, como han hecho aquí, por ejemplo:

Username	Password Hash (SHA256)
alice	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
bob	308738b8195da46d65c96f4ee3909032e27c818d8a079bccb5a1ef62e8daaa45
charlie	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

Pero entonces, ya que el hash no se puede descifrar, ¿Cómo comprobamos si la contraseña que mete el usuario cuando hace *log in* es correcta? Simplemente hasheamos la contraseña introducida.

Si tenemos al usuario **shacke** con el hash **5f4dcc3b5aa765d61d8327deb882cf99**:

- Si hace log in con la contraseña “password”
  - MD5(“password”) = **5f4dcc3b5aa765d61d8327deb882cf99**
  - Y por ende accederá a la página
- Si hace log in con otra contraseña
  - MD5(“test”) = **098f6bcd4621d373cade4e832627b4f6**
  - Como es diferente del hash que tenemos guardado accederá a la página

Cracking Hashes

Hemos dicho que un hash no se puede "descifrar" porque es irreversible. Pero sí podemos intentar **crackearlo**. Crackear, en este caso significa probar muchas combinaciones de contraseñas para intentar dar con la contraseña original.

Por ejemplo, como en el caso de antes, nuestro hash es

5f4dcc3b5aa765d61d8327deb882cf99

Si un atacante tiene una lista de contraseñas (que cree que son muy comunes), como por ejemplo esta:

hola

test

password

Este atacante puede probar a hashear todas esas contraseñas y comparar los hashes con el hash original. Este sería el funcionamiento:

MD5("hola") = 4d186321c1a7f0f354b297e8914ab240

MD5("test") = 098f6bcd4621d373cade4e832627b4f6

MD5("password") = **5f4dcc3b5aa765d61d8327deb882cf99**

Al llegar a la contraseña adecuada, los hashes son idénticos, y el atacante ha descubierto que la contraseña era "password". Esto se llama ataque por diccionario.

OSINT

Por último, vamos a ver qué es OSINT (Open-source intelligence). OSINT es una metodología de recolección, análisis y toma datos públicos para utilizarlos en un contexto de inteligencia.

Dicho de otro modo: vamos a usar Google para encontrar información pública muy útil sobre cualquier persona, empresa, servicio…

Google Dorking:

- Google Dorking consiste en utilizar Google de manera efectiva para filtrar solo la información útil que queremos encontrar

Google nos permite usar ciertos comandos en su buscador para filtrar. Por ejemplo, vamos a buscar documentos PDF en nuestra universidad, que contenga la palabra "malware".

Si buscamos "upm" en Google, vemos que hay **22,700,000** resultados, así que nos podríamos pasar la vida buscando y no encontrar nada.

Busquemos ahora “site:upm.es”. Hemos filtrado por el dominio de la UPM, y ahora solo tenemos **1,380,000** resultados. Pero siguen siendo muchos, y queremos encontrar PDFs.

Si ahora escribimos “site:upm.es filetype:pdf”, filtraremos para que exclusivamente nos aparezcan documentos PDF, y nos encontramos ahora con **199,000** resultados. Bastante bien.

Para acabar, escribimos “site:upm.es filetype:pdf “malware””. Al escribir la palabra clave entre comillas, Google buscará únicamente documentos que contengan esa palabra.

Hemos pasado de **22,700,000** a solo **669** resultados.

Estas técnicas se pueden utilizar para filtrar en redes sociales y buscar determinada información que nos pueda ser útil.

Lo vamos a dejar aquí por hoy, que solo de releerlo veo mucho texto y me dan nauseas.

Por ahora eso es todo, cualquier duda no dudéis en preguntar a cualquier compañero o a Llamas y Merk!

Hasta la semana que viene!