UNIVERSIDAD DEL VALLE DE GUATEMALA

Guatemala 23 de enero de 2024

Josúe Samuel Argueta Hernández - 211024

Ejercicio de Criptografía

Cifrado ROT: Historia y Aplicación

El cifrado **ROT** (rotación) es un método de sustitución que consiste en desplazar las letras del alfabeto un número fijo de posiciones. Este método pertenece a la categoría de cifrados clásicos y su variante más conocida, **ROT13**, desplaza las letras exactamente 13 posiciones.

Historia del cifrado ROT

El cifrado ROT tiene sus raíces en los cifrados por sustitución, siendo el más antiguo el **Cifrado César**, utilizado por el emperador romano Julio César para proteger comunicaciones militares confidenciales. Este cifrado funcionaba desplazando las letras del alfabeto un número determinado de posiciones, y es considerado uno de los primeros sistemas criptográficos de la historia (Singh, 2000).
En la actualidad, ROT13 se utiliza principalmente en aplicaciones triviales, como esconder spoilers en foros o transformar textos temporalmente en mensajes ilegibles a simple vista (Kahn, 1996).

Ejemplo de aplicación: Cifrado ROT13

"CRYPTOGRAPHY IS FUN".

1. **Alfabeto original**:
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
2. **Alfabeto cifrado (ROT13)**:
   NOPQRSTUVWXYZABCDEFGHIJKLM

Para cifrar, cada letra se reemplaza por la letra que está 13 posiciones adelante:
Texto claro: CRYPTOGRAPHY IS FUN
Texto cifrado: PELCBTCENCLV VF SHA
Para descifrar, se aplica el mismo desplazamiento. Debido a que el alfabeto tiene 26 letras, aplicar ROT13 dos veces devuelve el texto original.

Razones para elegir el cifrado ROT

Elegí el cifrado ROT porque desconocía como funcionaba, ahora veo que tiene sencillez, relevancia histórica y utilidad como herramienta educativa. Es una manera efectiva de enseñar conceptos básicos de criptografía y mostrar cómo la protección de la información ha evolucionado con el tiempo. En este caso, aunque sea muy simple, como he mencionado, si se puede apreciar cómo es que un cambio tan sutil, puede tener resultados totalmente diferentes y que pueden llegar a cumplir con el objetivo de ocultar información. Honestamente, no creo que se utilice tanto, al menos, no para el área empresarial de manera seria, pero si de manera educativa.

Ventajas y desventajas del cifrado ROT

Ventajas:

1. **Fácil de entender e implementar**: No se requiere de herramientas avanzadas ni conocimientos profundos.
2. **Simetría**: El mismo procedimiento sirve para cifrar y descifrar.
3. **Rapidez**: Se puede realizar manualmente o con algoritmos simples.
4. **Uso didáctico**: Es ideal para introducir los principios básicos de la criptografía.

Desventajas:

1. **Vulnerabilidad**: Es extremadamente fácil de romper. Por ejemplo, con ROT13 hay solo 25 desplazamientos posibles, lo que lo hace susceptible a ataques por fuerza bruta (Kahn, 1996).
2. **No oculta patrones**: Las frecuencias de las letras en el texto cifrado coinciden con las del texto claro, permitiendo su análisis y descifrado.
3. **Propósito limitado**: No es adecuado para proteger información sensible en la actualidad.

Conclusión

Aunque el cifrado ROT tiene poco valor práctico en entornos modernos, su simplicidad y contexto histórico lo convierten en una herramienta valiosa para la enseñanza de conceptos criptográficos básicos. Es un ejemplo de cómo la criptografía ha evolucionado desde métodos rudimentarios hasta los complejos algoritmos que se emplean en la actualidad.

EJERCICIO 2

Realizar la creación de un script que permita la conversión de palabras en texto **ASCII** a **BINARIO**

ascii_binary.py
Hola, mama respectivamente

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 ascii_binary.py
01001000 01101111 01101100 01100001
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 ascii_binary.py
01101101 01100001 01101101 01100001
```

Realizar la creación de un script que permita la conversión de palabras en texto **BASE64** a **BINARIO**

SG9sYQ==, 924sdfv

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 base64_binary.py
01001000011011110110110001100001000010000
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 base64_binary.py
11110111011011100010110001110101111110111111
```

Realizar la creación de un script que permita la conversión de **BINARIO** a **BASE64**

01001000011011110110110001100001000010000/01001000011011110110110001100001 => SG9sYQ==
11110111011011100010110001110101111110111111 => 924sdfv

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 binary_base64.py
SG9sYQ==
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 binary_base64.py
924sdfv=
```

Realizar la creación de un script que permita la conversión de **BINARIO** a **ASCII**

11110111011011100010110001110101111110111111 => ÷n,uû
01001000011011110110110001100001 => Hola

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 binay_ascii.py
÷n,uû
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 binay_ascii.py
Hola
```

Realizar la creación de un script que permita la conversión de **BASE64** a **ASCII**

924sdfv => ÷n,uû
SG9sYQ== => Hola

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 base64_ascii.py
÷n,uû
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 base64_ascii.py
Hola
```

Realizar la creación de un script que permita aplicar **XOR** a un **BINARIO**

10101010
11001100
01100110 ← resultado xor

01010101
11011100
10001001 ← resultado xor

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 xor_binary.py
01100110
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 xor_binary.py
10001001
```

Realizar la creación de un script que permita generar llaves dinámicas (utilizar **ASCII**)

longitud 10
jZvWYDyv0I

longitud 15
ZjP2lCNE7E0YfBK

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 dinamic_keys_ascii.py
jZvWYDyv0I
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 dinamic_keys_ascii.py
ZjP2lCNE7E0YfBK
```

Realizar la creación de un script que generar un nuevo cypher en **ASCII** con una llave **k de tamaño fijo**

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 newKey_fijo.py
)
Texto original: Hola
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$
```

Realizar la creación de un script que generar un nuevo cypher en **ASCII** con una llave **k de tamaño dinámico**

```
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$ python3 newKey_dinamic.py
PPZ
Texto original: HolaMundo
(env) arg@argher:~/Documents/uvg/cifrados_seguridad/ejercicios/e1$
```

Referencias

- Kahn, D. (1996). *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner.
- Singh, S. (2000). *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor.
- Stinson, D. R., & Paterson, M. B. (2019). *Cryptography: Theory and Practice*. CRC Press.
- https://chatgpt.com/share/6792dae1-3cfc-8010-88d8-083f78e5d8df