UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA, 2023-1

CRIPTOGRAFÍA

PRACTICAL SESSION 2

RC4

PROFESORA:

Dra. Rocío Aldeco Pérez

ALUMNA:

Karla Andrea Najera Noyola

Practical Session 2: RC4

Instrucciones

i. Describir como debe implementarse el algoritmo RC4 considerando la descripción presentada en el documento de la práctica.
ii. Describir el proceso a seguir para descifrar los datos.
iii. Subir un archivo con los hallazgos encontrados
iv. Implementar el proceso de cifrado y descifrado de RC4 mediante un lenguaje de programación, utilizando como referencia los siguientes casos de prueba:

Key	Keystream	Plaintext	Ciphertext
Key	EB9F7781B734CA72A719...	Plaintext	BBF316E8D940AF0AD3
Wiki	6044DB6D41B7...	pedia	1021BF0420
Secret	04D46B053CA87B59...	Attack at dawn	45A01F645FC35B383552544B9BF5

Introducción

Con el paso de los años la criptografía ha ido evolucionado con el fin de brindar soluciones cada vez más robustas, dando lugar a lo que hoy se conoce como la Criptografía Moderna. Esta rama fue creada en1948 a raíz de la Segunda Guerra Mundial y tomando como referencia la Teoría de la Información de Claude Shannon cuyo postulado principal era que ningún texto cifrado debía revelar ningún tipo de información acerca del texto sin formato, lo que implica que ningún atacante debería conocer información relacionada que sirva para su descifrado.

Dentro de esta clasificación de la criptografía, podemos encontrar los siguientes tipos:

- **Algoritmos asimétricos**: Son aquellos métodos que se basan en una codificación de información haciendo uso de 2 claves: una privada y una pública, con el fin de que el remitente conserve la clave privada y cualquier receptor pueda recibir la clave pública. Se les considera relativamente nuevos.
- **Algoritmos simétricos:** Son métodos más simples que los asimétricos debido a que utilizan la misma llave para las funciones de encriptación y desencriptación. Esta clave debe ser compartida con todas las personas que requieren recibir el mensaje.

En esta práctica nos enfocaremos en RC4, uno de los algoritmos simétricos más reconocidos en la actualidad, al cual implementaremos mediante código de un lenguaje de programación.

Marco Teórico

La presente investigación hará uso de los siguientes conceptos:

- **RC4**

RC4 es un algoritmo de cifrado de flujo diseñado por Ronald Rivest en 1987 para RSA Security. Fue (y sigue siendo) uno de los algoritmos más utilizados de esta categoría. Se consideraba secreto hasta que en el año de 1994 se filtró una descripción del algoritmo de manera anónima en una lista de correo de Cypherpunks tras lo cual pasó a ser publicado en numerosos sitios de Internet, pese a que hasta la fecha sigue siendo una marca registrada.

Tiene importancia debido a que es parte de los protocolos de cifrado más comunes para redes inalámbricas y se suele usar en aplicaciones como WEP de 1997 y WPA de 2003. También encontramos aplicaciones de RC4 en SSL de 1995 y es un sucesor de TLS de 1999.

Su éxito es en gran parte debido a que posee una alta velocidad y un método simple para su implementación tanto a nivel de software como de hardware.

- **Funciones XOR**

Una función XOR se basa en el uso de la compuerta XOR u OR exclusiva, la cual realiza una función booleana A'B+AB'. Es decir, se basa en el funcionamiento de la siguiente tabla:

Entrada A	Entrada B	Salida S
0	0	0
0	1	1
1	0	1
1	1	0

Cabe destacar que uno de sus usos comunes es la implementación de la adición binaria, además de como un comparador o inverso condicional. Asimismo, en el terreno de la criptografía permite la generación de números pseudoaleatorios, teniendo como ejemplo los registros de desplazamiento de realimentación lineal.

El símbolo por el cual se identifica esta función es un signo de más encerrado dentro de un círculo.

- **Secuencia pseudoaleatoria**

En primer lugar, un número pseudoaleatorio es aquel generado por algún proceso que aparente producir números al azar, pero que en realidad no es así. No obstante, no suelen tener un patrón aparente. Tomando en cuenta esto, una secuencia pseudoaleatoria se compone de varios números pseudoaleatorios y que pueden asemejarse al ruido, sin mostrar alguna periodicidad.

Hipótesis

Para comprender mejor este algoritmo, será necesario definir en primer lugar las 2 funciones en las que se basa su uso:

- ***Key-scheduling algorithm (KSA)***

Esta función es la encargada de inicializar la permutación en una matriz S a partir del uso de una clave que debe tener entre 1 a 255 caracteres. Su funcionamiento se basa en el siguiente pseudocódigo:

```
for i from 0 to 255
   S[i] := i
endfor
j := 0
for i from 0 to 255
   j := (j + S[i] + key[i mod keylength]) mod 256
   swap values of S[i] and S[j]
endfor
```

- **Pseudo-random generation algorithm (PRGA)**

Esta función se encarga de generar la secuencia final del algoritmo, o lo que se le conoce como keystream el cual representa a la cadena encriptada. Este valor se obtiene mediante intercambios de la matriz obtenida en el paso anterior además de la aplicación de algunas operaciones aritméticas. Para lograr su objetivo realiza los siguientes pasos:

```
i := 0
j := 0
while GeneratingOutput:
   i := (i + 1) mod 256
   j := (j + S[i]) mod 256
   swap values of S[i] and S[j]
   K := S[(S[i] + S[j]) mod 256]
   output K
endwhile
```

Una vez conocido lo anterior, una primera hipótesis podría mostrar que RC4 hace uso de las funciones anteriores siguiendo los pasos generales que presenta un algoritmo de flujo, en los cuales se combina el texto plano con un flujo de clave (keystream) generado a partir de una clave de cifrado y que se aplica sobre el texto plano dígito a dígito, así como del uso de la función XOR y secuencias pseudoaleatorias

En lo que respecta al proceso de descifrado, este será idéntico con respecto al cifrado. Aunque claramente debemos contar con la misma llave ya que, aunque tan solo haga falta un carácter o alguno sea erróneo, el resultado no será el adecuado.

Tomando en cuenta lo anterior, conoceremos los pasos generales del cifrado y descifrado con RC4, además de su implementación en código de Java.

Desarrollo de la solución

Propuesta del proceso de cifrado a partir de la implementación de las funciones ksa y prga

En primer lugar, se necesitan como valores de entrada por parte del usuario la clave y el texto a cifrar. Acto seguido, se obtiene la matriz S mediante la aplicación de la función ksa considerando la clave proporcionada. Una vez hecho este paso, se utiliza la función prga que hará uso de S y del texto a encriptar para la obtención del keystream. Al finalizar, se aplica un proceso de XOR entre el texto plano en formato de ascii y el flujo de datos del paso anterior, tras lo cual se presenta con sus respectivos valores hexadecimales. Estos pasos quedan ejemplificados de mejor forma con el siguiente pseudocódigo:

```
key=input()
plaintext=input()
S[]=ksa(key)
keystream=prga(s, plaintext)
xored_text = toAscii(plaintext) XOR keystream
hex_crypted=hex(xored_text)
```

Proceso de descifrado a partir del cifrado y del uso de la clave

Para el proceso de descifrado, se realizan básicamente los mismos pasos que para la encriptación debido al uso de la función XOR. En primer lugar, se obtienen los valores de la clave y el texto cifrado proporcioandos por parte del usuario. Cabe destacar que el texto cifrado primero debe tratarse de tal forma que se tengan sus valores decimales (siendo que originalmente estarán como una cadena hexadecimal). Se obtiene la matriz S mediante la aplicación de la función ksa considerando la clave proporcionada y una vez hecho este paso, se utiliza la función prga que hará uso de S y del texto a desencriptar para la obtención del keystream. Al finalizar, se aplica un proceso de XOR entre el texto cifrado y el flujo de datos del paso anterior, tras lo cual se presenta con los caracteres correspondientes que compondrán la cadena del mensaje desencriptado. Lo anterior se explica de mejor forma con el siguiente pseudocódigo:

```
key=input()
ciphertext=input()
S[]=ksa(key)
keystream=prga(s, ciphertext)
xored_text = toAscii(ciphertext) XOR keystream
decypt_text=(char)(xored_text)
```

Implementación del algoritmo en el lenguaje de programación Java

Se selecciono Java para la implementación de este algoritmo debido a la capacidad de uso que presentan las clases envolventes, además del manejo de cadenas y arreglos de texto/numéricos. Por ende, se usará el paradigma orientado a objetos.

Se hará uso de 2 clases por medio de las cuales un usuario podrá cifrar y/o descifrar sus mensajes:

- **Main.java:** Es la clase principal del proyecto. A través de esta se harán las interacciones con el usuario, incluyendo la captura de la clave y un mensaje, así como el cifrado/descifrado tras la aplicación del algoritmo. Se compone de un único método (Main) en el cual se imprime el menú de usuario y se realiza el flujo del programa.
- **Rc4.java:** Dentro de esta clase se definen todos los procesos y funciones que permiten el cifrado y descifrado de los mensajes que el usuario ingrese haciendo uso de los conceptos de RC4. Se compone de las siguientes 4 variables de instancia:
  - S[]: Representa el arreglo de 256 posiciones que se creará con ksa, con el fin de generar las permutaciones necesarias para el funcionamiento del algoritmo.
  - Keystream[]: Es un arreglo entero en el cual se almacenará el resultado obtenido tras la función prga con el fin de obtener el texto cifrado/descifrado.
  - key: Cadena que almacena la palabra clave ingresada por el usuario.
  - Texto: texto que deberá cifrarse o descifrarse según sea el caso.

  Asimismo, dentro de esta clase se encuentran los siguientes métodos:
  - Rc4(): Es el constructor de la clase. Se encarga de recibir la clave y el texto proporcionados por el usuario.
  - ksa(): Equivalente en código de la función ksa mostrada en pseudocódigo en secciones anteriores. En primer lugar, obtiene la longitud de la clave y llena

las 255 posiciones del vector con valores consecutivos. Acto seguido, realiza la mezcla de bytes en la matriz realizando los intercambios correspondientes.

- prga(): Implementación de la función prga mostrada anteriormente. Genera la salida a partir de los intercambios correspondientes y de la aplicación de funciones aritméticas simples (como módulo y sumas). Cabe destacar que recibe el valor de longitud debido a que no se puede usar el mismo para encriptado y desencriptado (desencriptado debe tomar solo la mitad del tamaño debido a que el texto se presenta con valores hexadecimales que ocupan 2 posiciones).
- encriptar(): Aplica las instrucciones necesarias que permiten transformar una cadena mediante RC4. En resumen, implica la transformación de texto a decimales ASCII, la aplicación de ksa y prga (con la longitud completa del texto), la aplicación de XOR entre el keystream y el texto en decimales ASCII, así como la correspondiente conversión a valores hexadecimales. Como valor de salida devuelve la cadena encriptada.
- desencriptar(): Prácticamente realiza los mismos pasos que en la función anterior, con la diferencia de que se considera solo la mitad de la longitud del texto dado que cada valor en hexadecimal utiliza 2 posiciones. En primer lugar, convierte cada subcadena hexadecimal a números enteros para aplicar ksa y prga. Acto seguido, realiza XOR entre el keystream y el texto en decimal, así como su correspondiente conversión a caracteres, la cual se devuelve como un String con el mensaje desencriptado.

Para la ejecución de este programa, se implementa un menú con las siguientes 3 opciones:

1. **Encrypt**: Al seleccionar esta opción, se realiza la encriptación de un mensaje que un usuario brinde junto con su clave correspondiente.
2. **Decrypt**: Al seleccionar esta opción, se realiza el descifrado de un mensaje dado a partir del algoritmo propuesto, haciendo uso de la clave y texto proporcionados por el usuario.
3. **Exit**: Termina la ejecución del programa.

Cabe destacar que el programa se repite de manera infinita hasta que el usuario seleccione la opción de “Exit”.

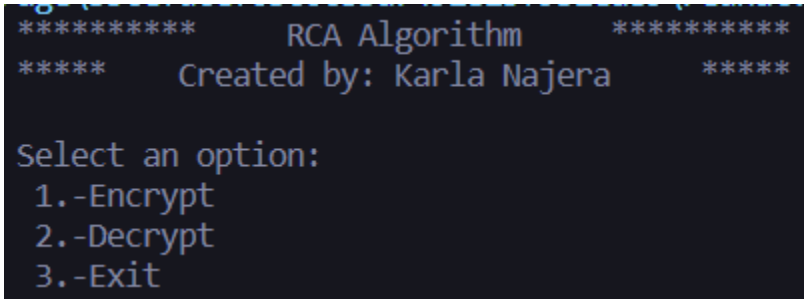
Aunque las clases ya han sido compiladas, en caso de ser requerido se puede compilar con la siguiente instrucción:

javac Main.java

Y se ejecuta con el siguiente comando:

java Main.java

Nota: Para su compilación es necesario tener instalada alguna versión de JDK, aunque la versión que fue utilizada para su elaboración es la 18.0.2.1.



A continuación, se muestran algunas capturas de pantalla de su funcionamiento tomando como referencia los vectores de prueba proporcionados para esta práctica

Llave y Texto Plano	Cifrado	Descifrado
Key **Plaintext**	Key: Key Plaintext: Plaintext Encrypted Message: BBF316E8D940AF0AD3	Key: Key Encrypted Text: BBF316E8D940AF0AD3 Decrypted Message: Plaintext
Wiki **pedia**	Key: Wiki Plaintext: pedia Encrypted Message: 1021BF0420	Key: Wiki Encrypted Text: 1021BF0420 Decrypted Message: pedia
Secret **Attack at dawn**	Key: Secret Plaintext: Attack at dawn Encrypted Message: 45A01F645FC35B383552544B9BF5	Key: Secret Encrypted Text: 45A01F645FC35B383552544B9BF5 Decrypted Message: Attack at dawn

Aunque se adjunta un archivo en zip con los archivos utilizados para esta práctica, se comparte el enlace de GitHub: https://github.com/kiliapplered/Practica2Cripto.git

Conclusiones

Del presente trabajo es posible obtener las siguientes conclusiones:

- A partir de la presente práctica fue posible conocer uno de los algoritmos más famosos de cifrado de flujo, siendo en este caso el algoritmo de RC4.
- La hipótesis presentada para esta práctica fue correcta, siendo que se utilizaron los elementos básicos de un algoritmo de cifrado de flujo, como lo son el uso de una llave y funciones XOR.
- Aunque hoy en día podría resultar simple el funcionamiento de este algoritmo e incluso cuestionable debido a que en la actualidad se encuentra “roto”, no quedan dudas que en

su momento presentó un avance significativo con respecto a lo que hoy conocemos como Criptografía Moderna

- La implementación de este algoritmo como una solución "compleja y segura" no es deseable en estos días, sin importar que en algunos lados se siga usando como tal. En su lugar debe optarse por soluciones con un mayor nivel de complejidad y de los cuales no se conozca el algoritmo (o este sea mucho más difícil de descifrar).
- La realización de este ejercicio permitió solidificar los conocimientos adquiridos durante la revisión teórica de la asignatura.

Referencias

[1] UPM, «Píldora formativa 35: ¿Cómo funciona el algoritmo RC4?,» 2016.

[2] «Acervo Lima,» ¿QUÉ ES EL CIFRADO RC4?, [En línea]. Available: https://es.acervolima.com/que-es-el-cifrado-rc4/. [Último acceso: 24 Septiembre 2022].

[3] Geeks for Geeks, «Geeks for Geeks,» Implementation of RC4 algorithm, 9 Agosto 2021. [En línea]. Available: https://www.geeksforgeeks.org/implementation-of-rc4-algorithm/#:~:text=RC4%20is%20a%20symmetric%20stream,with%20the%20generated%20key%20sequence.. [Último acceso: 24 Septiembre 2022].

[4] A. Boya García, «El cifrado RC4,» Universidad de Salamanca, [En línea]. Available: https://ntrrgc.me/attachments/Cifrado_RC4/. [Último acceso: 24 Septiembre 2022].

[5] D. E. Bernal Michelena, «LA CRIPTOGRAFÍA: EL SECRETO DE LAS COMUNICACIONES SEGURAS,» *Revista Seguridad UNAM,* nº 11, 2016.

[6] J. J. Meneu, «Detalles sobre la criptografía moderna,» Arrow, 18 Mayo 2016. [En línea]. Available: https://www.arrow.com/es-mx/research-and-events/articles/modern-cryptography. [Último acceso: 20 Septiembre 2022].

Yo, Karla Andrea Najera Noyola, hago mención que esta práctica fue de mi autoría.