UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA, 2023-1

CRIPTOGRAFÍA

Practical Session 4

MD2

PROFESORA:

Dra. Rocío Aldeco Pérez

ALUMNA:

Karla Andrea Najera Noyola

MD2

Instrucciones

a. Utilizando la descripción y el pseudocódigo provistos, discuta cómo se debe implementar este algoritmo, cuál sería el mejor lenguaje de programación para hacerlo y crear un borrador de los elementos que compondrían al código.
b. Subir un archivo con todos estos hallazgos (individual).
c. Realice un envío individual de su implementación utilizando el lenguaje de programación de su elección. Los casos de prueba son los presentados en la siguiente tabla:

MD2 test suite	
Plaintext	**Hash**
("")	8350e5a3e24c153df2275c9f80692773
("a")	32ec01ec4a6dac72c0ab96fb34c0b5d1
("abc")	da853b0d3f88d99b30283a69e6ded6bb
("message digest")	ab4f496bfb2a530b219ff33031fe06b0
("abcdefghijklmnopqrstuvwxyz")	4e8ddff3650292ab5a4108c3aa47940b
("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")	da33def2a42df13975352846c30338cd
("12345678901234567890123456789012345678901234567890123456789012345678901234567890")	d5976f79d83d3a0dc9806c3c66f3efd8

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.

Hoy en día se ha vuelto crucial el uso de algoritmos que permitan la transferencia segura de datos a través de canales que no siempre pueden tener las máximas medidas para garantizar que la información cumpla con la triada de la seguridad. Por ello, es cada vez más frecuente la aparición de nuevos algoritmos con un mayor nivel de complejidad. Tomando en cuenta lo anterior, dentro de la Criptografía Moderna existirá la siguiente clasificación:

- **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 la actualidad, algunos de los algoritmos asimétricos requieren de funciones especiales adicionales denominadas como hash que garanticen que la información enviada sea de una sola vía. Por ello, en esta práctica nos enfocaremos en el funcionamiento de MD2, uno de los algoritmos de la familia MD (Message Digest) que tras algunas actualizaciones sentaron las bases de lo que hoy en día es MD5, uno de los algoritmos de reducción criptográficas más utilizados en la actualidad.

Marco Teórico

La presente práctica hará uso de los siguientes conceptos:

- **Función Hash**

Son funciones encargadas de generar una cadena de bits a partir de un conjunto de datos dado. No importando el tamaño de la cadena de datos, su objetivo es (idealmente) generar una cadena de longitud fija y menor al valor de entrada, además de que, sin importar el número de veces que se ejecute el algoritmo, siempre debe arrojar la misma salida (a menos que se cambie al menos 1 bit de la cadena original).

Consisten en funciones irreversibles, por lo que a partir de la salida generada no debe ser posible recuperar el conjunto de datos original. De igual forma, deben ser resistentes a colisiones, las cuales pueden llegar a ser muy frecuentes.

Son ampliamente utilizadas para verificar la autenticidad de un mensaje debido a su corto tamaño, lo que a su vez implica que deben cumplir con los servicios de seguridad de integridad y confidencialidad.

También se les conoce como funciones resumen o *digest*, debido a sus salidas breves en comparación de los datos originales que fueron cifrados.

- **Message Digest**

Un resumen de mensaje es una representación numérica de tamaño fijo del contenido de un mensaje calculado a partir de funciones hash. Debe cumplir con ser unidireccional y, en la medida de lo posible, computacionalmente inviable que existan 2 mensajes con un mismo

resumen. Los algoritmos más famosos de este tipo (MD2, MD4, MD5 y MD6) fueron diseñados por Ronald Rivest a partir de los años 90 (mismo creador de RSA).

- **S-Box**

Es uno de los componentes fundamentales en los algoritmos de cifrado de clave simétrica y en el caso de los cifradores por bloque (como es el caso de DES o AES) se utilizan para proteger y realizar la correspondencia entre un texto plano y uno cifrado. Suelen ser elegidas de tal forma que resistan el criptoanálisis y garanticen la seguridad de los mensajes.

Por lo general, las S-Box cuentan con una entrada de m bits y una salida de n bits haciendo uso de técnicas de sustitución

- **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.

- **Colisión**

Es un suceso en el que 2 valores de entrada diferentes generan un mismo resumen o *digest* tras su procesamiento por alguna función hash. Aunque en principio no deberían existir, a lo largo de la historia se han demostrado apariciones frecuentes en variados algoritmos, lo que impide el cumplimiento de la confidencialidad, autenticidad e integridad en datos enviados a través de un medio de transmisión. La solución ante este concepto es la creación de algoritmos más seguros que generen salidas más extensas (256 o incluso 512 bits), siendo algunos de ellos el caso de SHA-256 Y SHA-512.

Hipótesis

Previo a una investigación más profunda acerca del funcionamiento de MD2 supondremos que este algoritmo, a tratarse de una función de hash criptográfico, hará uso de permutaciones con una S-Box (que está previamente definida). Asimismo, dado que existen versiones posteriores del algoritmo supondremos que funciona con una longitud de bloque considerablemente chica y que resulta en posibles colisiones en algunos datos, así como fallos en su seguridad.

Tomando en cuenta lo anterior, a partir de la resolución de la práctica demostraremos lo planteado como hipótesis al plantear el código en un lenguaje de programación.

Desarrollo

¿Qué es MD2? ¿Cuáles son sus principales características

MD2 es el algoritmo de Message Digest Algorithm 2, el cual es una función de hash criptográfica desarrollada por Ronald Rivest en 1989. Este algoritmo toma como valor de entrada un mensaje con longitud arbitraria y a partir de una serie de operaciones (incluido el uso de XOR) produce una "huella digital" o resumen de mensaje (digest) de 128 bits, o lo que es equivalente a 16 bytes.

Este algoritmo se encuentra diseñado para aplicaciones que hagan uso de la firma digital, donde es una necesidad la compresión de archivos de gran tamaño antes de firmarse con una clave privada bajo un criptosistema de clave pública (como es el caso de RSA).

Aunque originalmente estaba documentado en el RFC 1319, presenta una actualización en el RFC 1115 donde se corrige una asignación y se agrega un XOR que beneficia a los mensajes con una extensión grande (más de 16 bytes o caracteres).

Su funcionamiento es fácil de describir a partir del uso de 3 funciones fundamentales:

1. Padding: Se completa el mensaje de entrada de tal forma que se obtenga un múltiplo de 16 bytes como tamaño de la cadena.
2. Checksum: Se agrega una suma de comprobación de 16 bytes al mensaje obtenido en el paso anterior.
3. Hash: Se realiza el proceso de hash contando con un arreglo de 48 bytes inicializado en ceros y, en bloques de 16 bytes, se procesan las cadenas de tal forma que sea posible generan el resumen (digest) del mensaje original.

Borrador del código

Tomando en cuenta el funcionamiento general del algoritmo se realizará una propuesta de pseudocódigo para cada una de sus 3 partes fundamentales.

En el caso del paso 1, Padding, su pseudocódigo es el siguiente:

```
Padding = tam_bloque-(longMensaje%tam_bloque)
for (i=0; i<padding; i++) {
   mensaje+=padding
}
```

Para el paso 2, Checksum, el procedimiento general es el siguiente:

```
for(i=0; i<longMensaje/16; i++) {
   for(j=0; j<16; j++) {
      bytes= mensaje[i*16+j]
      checksum[j] ^= sBox[bytes^checkbyte_previo]
      checkbyte_previo=checksum[j]
   }
}
mensaje+=checksum
```

Por último, para el paso 3, Hash, el pseudocódigo sería el siguiente:

```
for(i=0; i<longMensaje/16; i++) {
   for(j=0; j<16; j++) {
      resumen[16+j]=mensaje[i*16+j]
      resumen[32+j]= resumen[tam_bloque+j]^resumen[j]
   }
   check=0
   for(j=0; j<18; j++) {
      for(k=0; k<48; k++){
         check=resumen[k]^sBox[check]
          resumen[k]=check
      }
      check=(check+j)%256
   }
}
mensaje_final=resumen[0-15]
```

Cabe destacar que, para cada una de las funciones, el valor de la longitud del mensaje se irá actualizando, siendo que se suman los valores del padding, así como el del checksum, por lo que, pese a que podría parecer que se indica de dicha forma, el valor no es estático.

Asimismo, pese a que se tiene como valor de entrada una cadena de texto, en realidad trabajaremos con el valor de ASCII de los caracteres introducidos, por lo que se optará por emplear listas ligadas de valores enteros.

Por último, como una de las consideraciones de mayor importancia que debemos tomar en cuenta es que la S-Box con la que trabajaremos para las sustituciones incluirán decimales de Pi, los cuales son los siguientes:

```
int sBox[]= {
        41, 46, 67, 201, 162, 216, 124, 1, 61, 54, 84, 161, 236, 240, 6, 19,
        98, 167, 5, 243, 192, 199, 115, 140, 152, 147, 43, 217, 188, 76, 130, 202,
        30, 155, 87, 60, 253, 212, 224, 22, 103, 66, 111, 24, 138, 23, 229, 18,
        190, 78, 196, 214, 218, 158, 222, 73, 160, 251, 245, 142, 187, 47, 238, 122,
        169, 104, 121, 145, 21, 178, 7, 63, 148, 194, 16, 137, 11, 34, 95, 33,
        128, 127, 93, 154, 90, 144, 50, 39, 53, 62, 204, 231, 191, 247, 151, 3,
        255, 25, 48, 179, 72, 165, 181, 209, 215, 94, 146, 42, 172, 86, 170, 198,
        79, 184, 56, 210, 150, 164, 125, 182, 118, 252, 107, 226, 156, 116, 4, 241,
        69, 157, 112, 89, 100, 113, 135, 32, 134, 91, 207, 101, 230, 45, 168, 2,
        27, 96, 37, 173, 174, 176, 185, 246, 28, 70, 97, 105, 52, 64, 126, 15,
        85, 71, 163, 35, 221, 81, 175, 58, 195, 92, 249, 206, 186, 197, 234, 38,
        44, 83, 13, 110, 133, 40, 132, 9, 211, 223, 205, 244, 65, 129, 77, 82,
        106, 220, 55, 200, 108, 193, 171, 250, 36, 225, 123, 8, 12, 189, 177, 74,
        120, 136, 149, 139, 227, 99, 232, 109, 233, 203, 213, 254, 59, 0, 29, 57,
        242, 239, 183, 14, 102, 88, 208, 228, 166, 119, 114, 248, 235, 117, 75, 10,
        49, 68, 80, 180, 143, 237, 31, 26, 219, 153, 141, 51, 159, 17, 131, 20};
```

Cabe destacar que estos valores son dados por el RFC 1319 como default.

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 (especialmente Integer), además del manejo de arreglos numéricos (que inicializan todas las posiciones con ceros) y las listas de enteros. Por ende, se usará el paradigma orientado a objetos.

Para la aplicación del algoritmo se utilizará una única clase conocida como Main.java en la cual se le solicitará al usuario el mensaje al que se le aplicará el proceso de hash y se devolverá el resultado obtenido tras el uso de los pasos de MD2.

En esta clase, en primer lugar, se imprimirán datos de bienvenida al programa y se solicitará al usuario la inserción del mensaje la cual se almacenará en un String. Acto seguido, se convertirá dicha cadena en una lista de enteros con la representación en ASCII, lo cual es fundamental para el desarrollo de los pasos siguientes.

Una vez que se tiene el mensaje con el formato solicitado, se inicia al algoritmo con el paso 1 de MD2 (Padding) donde se incrementará el tamaño del mensaje con el fin de que sea un múltiplo de 16 bytes agregando “i” bytes de valor “i”. Acto seguido, se agrega el checksum, o bien, la suma

de comprobación de 16 bytes al mensaje obtenido en el paso anterior procesando bloques de 16 bytes y aplicando operaciones XOR acorde a lo establecido en la actualización del RFC 1319. Al final de este paso, se añaden uno a uno los caracteres generados al mensaje procesado.

Como último paso se realiza el hash, lo que genera como resultado el digest del mensaje a partir de ciclos anidados que procesan el mensaje en bloques de 16 bytes y por 18 rondas a un tamaño de buffer de 48 bytes. Para finalizar la aplicación del algoritmo se utilizan los primeros 16 bytes del digest generado y se representan con sus valores en hexadecimal, imprimiendo esto al usuario y terminando con ello la ejecución del programa.

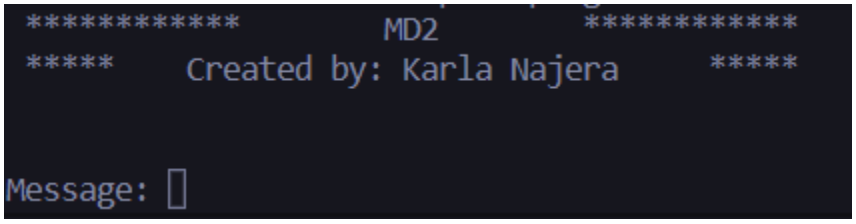
Aunque las clases ya han sido compiladas, en caso de ser requerido se puede compilar con la siguiente instrucción:

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

Texto Plano	Hash obtenido con el programa
("")	Message: Hash: 8350e5a3e24c153df2275c9f80692773
("a")	Message: a Hash: 32ec01ec4a6dac72c0ab96fb34c0b5d1
("abc")	Message: abc Hash: da853b0d3f88d99b30283a69e6ded6bb
("message digest")	Message: message digest Hash: ab4f496bfb2a530b219ff33031fe06b0
("abcdefghijklmnopqrstuvwxyz")	Message: abcdefghijklmnopqrstuvwxyz Hash: 4e8ddff3650292ab5a4108c3aa47940b
("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")	Message: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 Hash: da33def2a42df13975352846c30338cd
("12345678901234567890123456789012345678901234567890123456789012345678901234567890")	Message: 12345678901234567890123456789012345678901234567890123456789012345678901234567890 Hash: d5976f79d83d3a0dc9806c3c66f3efd8

Aunque se adjunta un archivo en zip con los archivos utilizados para esta práctica, se comparte el enlace de GitHub: https://github.com/kiliapplered/Practica4Cripto.git

Conclusiones

Del presente trabajo es posible obtener las siguientes conclusiones:

- A partir de la presente práctica fue posible conocer el algoritmo MD2, el cual es una función de hash criptográfica que es un antecesor directo de MD5, uno de los métodos más conocidos en la actualidad.
- La hipótesis presentada para esta práctica fue correcta, siendo que hace uso de permutaciones con una S-Box previamente establecida, con una longitud de bloque bastante chica y que incluso presenta problemas de colisiones.
- Hoy en día podría resultar simple el funcionamiento de este algoritmo; no obstante, no quedan dudas que en su momento presentó un avance significativo con respecto a lo que hoy conocemos como Criptografía Moderna y que dio pie a todas las evoluciones de la familia Message Digest.
- La implementación de este algoritmo como una solución "compleja y segura" no es deseable en estos días e incluso OpenSSL y GnuTLS desactivaron esta función hash en 2009 por sus problemas de colisiones.
- En su lugar (e incluso de algunas de sus actualizaciones) debe optarse por soluciones con un mayor nivel de complejidad como los son los algoritmos de la familia SHA.
- La realización de este ejercicio permitió solidificar los conocimientos adquiridos durante la revisión teórica de la asignatura.

Referencias

[1] 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].

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

[3] R. KeepCoding, «¿Qué es una colisión en criptografía?,» Keepcoding, 20 Julio 2022. [En línea]. Available: https://keepcoding.io/blog/que-es-una-colision-en-criptografia/. [Último acceso: 1 Noviembre 2022].

[4] B. Kaliski, «The MD2 Message-Digest Algorithm,» Datatracker, Abril 1992. [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc1319. [Último acceso: 4 Noviembre 2022].

[5] «RFC 1319, "The MD2 Message-Digest Algorithm", April 1992,» RFC Editor, 11 Febrero 2002. [En línea]. Available: https://www.rfc-editor.org/errata/eid555. [Último acceso: 5 Noviembre 2022].

[6] «MD2 - algoritmo resumen,» DIT UPM, [En línea]. Available: http://www.dit.upm.es/~pepe/401/5273.htm#!-alone.

[7] N. F., «Cryptography hash method MD2 (Message Digest 2) explained with Python,» Nick from The Crypt - Medium, 17 Noviembre 2019. [En línea]. Available: https://nickthecrypt.medium.com/cryptography-hash-method-md2-message-digest-2-step-by-step-explanation-made-easy-with-python-10faa2e35e85. [Último acceso: 4 Noviembre 2022].

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