FUNDAMENTOS BÁSICOS

I. Operaciones básicas en matemáticas con números pequeños

1.1. Desde la carpeta SAMCript1.0 ejecuta el programa SamCript.jar.

1.2. Para familiarizarte con el programa, por favor haz primero un recorrido por su menú y recuerda que en él tienes un Manual de Usuario y un Banco de Pruebas.

Nota: ten la precaución de comprobar antes de cada operación en qué unidades estás trabajando. Si los números son pequeños, se puede confundir una unidad decimal con una hexadecimal, e.g. el valor 50941 puede ser decimal o hexadecimal.

1.3. comprueba los siguientes resultados eligiendo las unidades Decimal, Binario y Hexadecimal según corresponda. Observa las salidas en modo hexadecimal y en binario.

1.3.1. 24.368.520.987.296 + 62.076.027.571.982 = 86.444.548.559.278.

1.3.2. 1000110011 + 1111010010 = 11000000101.

1.3.3. ACDC + ABBA = 15896 (comprueba manualmente este resultado).

1.3.4. 398.073.288 - 591.983.001 = -193.909.713.

1.3.5. 876.080.271.128.375 x 563.910.654.553 = 494.030.999.132.971.654.141.241.375.

1.3.6. 101100 x 111001010 = 100111010111000.

1.3.7. A1B2C3 x D4E5F7 = 86794A67E925.

1.3.8. 874.576.357.263.529.872.652 / 8.993.686.876.376.389.872 = 97,243362959497.

1.3.9. 1110010011000101111101 / 1001000101101010101 = 1100.

II. Operaciones modulares básicas: mod, suma, resta, multiplicación, xor Ejercicio 2)

2.1.2. BEBE mod ABBA = 1304.

2.1.3. 10010010100101001011 mod 10010011110101011001 = 10010010100101001011.

2.1.4. 91 + 123 mod 100 = 14.

2.1.5. BECA + ACABA mod ACDC = EE8.

2.1.6. 10001011 + 10011000 mod 111 = 100.

2.1.7. 851.876 - 1.513 mod 77 = 52.

2.1.8. 80.128.375 x 63.421 mod 127 = 20.

III Criptografía clásica

3.1. Usando el programa Criptoclasicosv2.1.jar y usando el archivo QUIJOTE MANCHA.txt, realice las siguientes operaciones, usando como lenguaje español Z27:

3.1.1. Cifre, usando el desplazamiento puro (CESA) con la clave K = 3.

3.1.2. Cifre, usando el desplazamiento puro (CESA) con la clave K = 5.

3.1.3. Cifre, usando el desplazamiento puro (CESA) con la clave K = 25.

3.2. Usando el programa Criptoclasicosv2.1.jar y usando el archivo QUIJOTE MANCHA.txt, realice las siguientes operaciones, pero esta vez usando el lenguaje Ingles Z26:

3.2.1. Cifre, usando el desplazamiento puro (CESA) con la clave K = 3.

3.2.2. Cifre, usando el desplazamiento puro (CESA) con la clave K = 5.

3.2.3. Cifre, usando el desplazamiento puro (CESA) con la clave K = 25.

3.3. Compare los 6 cifrados con claves similares (3.1 y 3.2). Se pueden ver/hallar alguna similitud/ diferenciar hay entre estos. Puede explicar matemáticamente ¿por qué ocurre?

3.4. Usando el programa Criptoclasicosv2.1.jar y tomo al azar cualquier cifrado en los puntos anteriores y realiza un análisis de criptoanálisis. Justifique que ocurre y ¿por qué?.

3.5. Usando el programa Criptoclasicosv2.1.jar y usando el archivo QUIJOTE MANCHA.txt, cifre usando Vigenerè, utilice una clave de 4 y 10 palabra tanto el lenguaje español Z27 como el lenguaje Ingles Z26. Hay diferencias, (si /no) ¿por qué?

3.6. Usando el programa Criptoclasicosv2.1.jar cualquiera de los cifrado del 3.5 y realizando criptoanálisis con Vigenerè. En que se modifica que se utilice tanto el lenguaje español Z27 como el lenguaje Ingles Z26. ¿Hay diferencias, por qué?

3.7. Usando el programa Criptoclasicosv2.1.jar y usando el archivo QUIJOTE MANCHA.txt, cifre usando trasposición filas, utilice una clave menor a 10 y una mayor a 10. Usando tanto el lenguaje español Z27 como el lenguaje Ingles Z26. Si cripto analiza los resultados, ¿qué diferencias hay? con el tamaño de las filas, o con el idioma. Justifique su respuesta.

3.8. Repita el 3.7 pero esta vez usando la trasposición columna.

IV CIFRADO SIMÉTRICO EN BLOQUES – DES. Comprobación del funcionamiento de DES en modo ECB, usando el Software safeDES

4.1 Con la clave ASCII K = **123** Clave cifre el mensaje 8 bytes M = **Hola Ana.**

- 4.1.1 Usando el portapapeles, descifre el criptograma y observe si hay relleno.
- 4.1.2 Repita la cifra del mensaje usando ahora K = **A9A83CFA8B16CF0D** una clave hexadecimal y compruebe nuevamente si hay relleno.

4.2 Utilizando las dos claves del punto anterior, cifre el mensaje M = **No me parece importante el largo.**

- 4.2.1 Compruebe si ahora hay un relleno de un byte para formar un segundo bloque de texto en hexadecimal de 64 bits.

4.3 Cifre el mensaje M = **Ya no te saludo más Lucía.**

- 4.3.1. Con la clave hexadecimal K = **1111111111111111**
- 4.3.2. Con la clave hexadecimal K = **0000000000000000**
- 4.3.3. Con la clave ASCII K = **BCBCBCBC**
- 4.3.4. Con la clave K = **BCCCCCCB**.
- 4.3.5 Compare el 4.3.3 y 4.3.4. Explique y justifique lo que ha sucedido.
- 4.3.6 Repita la cifra ahora con las claves hexadecimal K = **1111112222111111** y K = **2222221111222222**. Explique y justifique lo que ha sucedido.

4.4 Cifre el mensaje M = **Probaremos un ataque por fuerza bruta**, con la clave K = **AAABBAAA**.

- 4.4.1 Abra una nueva ventana y descífrelo usando el portapapeles y tomando como entrada el criptograma en hexadecimal.
- 4.4.2 Con ambos textos (claro y criptograma) en hexadecimal proceda a un ataque monousuario con clave inicial **AAABA000** y clave final **AAABBDDD**.
- 4.4.3 ¿Qué puede decir con respecto al tiempo de criptoanálisis real y el tiempo que se requeriría para recorrer todo el espacio de claves dado?.
- 4.4.4 Comente y justifique el número de claves válidas encontradas.

4.5 Repita el ataque con clave inicial **AAABB777** y final **AAABFFFF**. Comente lo que ha sucedido con respecto al espacio de claves elegido.

4.6 Cifre el mensaje M = **Probaremos un ataque por fuerza bruta**, pero usando un ataque de simulación multiusuario con la clave inicial **AAABB000** y la clave final **AAABBDDD**. Elija el número de procesos desde 1 hasta 10 y observe lo que sucede con el tiempo de ataque a la clave. Justifique lo que ha visto.

4.7 Repita el ejemplo anterior usando un ataque de simulación multiusuario con la misma clave inicial **AAABB000** pero una clave final **AAABBFFF**.

4.8 Cifre el mensaje M = **Ahora atacaremos claves en hexadecimal**,

4.8.1 con la clave K = **1111222233334444**. La clave inicial es **111122223327BF6F** y la clave final **11112222333B72EE**.

4.8.2 Proceda al ataque monousuario.

4.8.3 Con la calculadora de Windows en hexadecimal reste la clave inicial de la clave final y encuentre luego su valor en decimal.

4.8.4 ¿Por qué no coincide este valor con el número de claves distintas que indica el programa?

V CIFRADO SIMÉTRICO EN BLOQUES – DES. Comprobación del funcionamiento de DES en modo CBC, usando el Software safeDES

5.1 Repita 4.1 y 4.2 usando DES en modo CBC.

5.2 ¿Cuándo puede usar el modo de cifrado ECB? Justifique.

VI CIFRADO SIMÉTRICO EN BLOQUES – TripleDES. Usando el Software safeDES

6 Use cualquiera de los ejemplo y pruebe ¿Por qué se usa en el triple DES un cifrado con sólo dos claves tipo EDE y no con tres como su nombre indica?

VII CIFRADO SIMÉTRICO EN BLOQUES -AES. Complete el enunciado con el Software AESPHERE en modo ECB

Todos estos valores y muchos otros vectores de validación de AES puedes encontrarlos en la sección "Test Vectors for AES" de la página oficial del NIST, abriendo el archivo KAT_AES.zip cuya descarga se recomienda para su correspondiente comprobación.

Web NIST Test Vectors for AES: **http://csrc.nist.gov/groups/STM/cavp/index.html**
https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/aes/KAT_AES.zip

7.1 Con la opción comprobar vectores en formato hexadecimal de AESphere que verás en la parte superior derecha de página principal o en el menú, cifra el texto M con la clave K de 128, 192 y 256 bits que se entregan. Observa que se obtienen los criptogramas C que se indican y que son los correctos según la página anterior del NIST de la carpeta KAT_AES que has descargado.

M = 00000000000000000000000000000000

7.1.1 Si K1 = 10a58869d74be5a374cf867cfb473859
C1 = 6d251e6944b051e04eaa6fb4dbf78465

7.1.2 Si K2 = e9f065d7c13573587f7875357dfbb16c53489f6a4bd0f7cd
C2 = 0956259c9cd5cfd0181cca53380cde06

7.1.3 Si K3 =c47b0294dbbbee0fec4757f22ffeee3587ca4730c3d33b691df38bab076bc558
C3 = 46f2fb342d6f0ab477476fc501242c5f

7.2 Proceso de descifrado. Al igual que en el apartado anterior, nuevamente con la opción comprobar vectores en formato hexadecimal, descifra estos criptogramas C con sus correspondientes claves K de 128,192 y 256 bits, y observa que se obtiene siempre el mismo texto en claro M indicado con cadenas de ceros y que son los que se indican en esa carpeta KAT_AES.

M = 00000000000000000000000000000000

7.2.1 Si C1 = 6d251e6944b051e04eaa6fb4dbf78465
K1 = 10a58869d74be5a374cf867cfb473859

7.2.2 Si C2 = 8e4e18424e591a3d5b6f0876f16f8594
K2 = 15d20f6ebc7e649fd95b76b107e6daba967c8a9484797f29

7.2.3 C3 = 46f2fb342d6f0ab477476fc501242c5f
K3 = c47b0294dbbbee0fec4757f22ffeee3587ca4730c3d33b691df38bab076bc558

Ejercicio: busca en la carpeta KAT_AES que has descargado los archivos que han sido usados en los dos ejercicios anteriores.

VIII CIFRADO SIMÉTRICO EN BLOQUES -AES. Complete el enunciado con el Software AESPHERE en modo CBC

Desde la misma página oficial del NIST (archivo CBCGFSbox128.rsp) comprueba desde la aplicación AESphere con la opción Cifrar en modo CBC que los primeros 128 bits del criptograma se corresponden con lo que aparece en dicho documento y que se muestra a continuación:

a) Proceso de cifrado
M = f34481ec3cc627bacd5dc3fb08f273e6
K = 00000000000000000000000000000000
IV = 00000000000000000000000000000000
C = 0336763e966d92595a567cc9ce537f5e
Al cifrar, AESphere te mostrará:
M = f34481ec3cc627bacd5dc3fb08f273e610101010101010101010101010101010 C = 0336763e966d92595a567cc9ce537f5ef1a80d711806421a109873def502863e

Los primeros 32 valores en hexadecimal de C subrayados son los indicados por el documento del NIST. Veremos a continuación el significado de esos 16 valores hexadecimales 10 que observas en el mensaje M.

IX CIFRADO SIMÉTRICO EN BLOQUES -AES. Complete el enunciado con el Software AESPHERE en Relleno en AESphere

El algoritmo AES trabaja sobre un tamaño fijo de bloque de 128 bits. Como es lógico, la aplicación AESphere permite cifrar textos o documentos grandes, textos que no tienen por qué coincidir exactamente con el tamaño de bloque. Nos encontramos entonces ante textos que deben ser rellenados para poder ser cifrados, como sucede en todos los cifradores de bloque.
Existen distintos métodos de rellenado. El que se usa en AESphere es uno de los más comunes y se basa en rellenar con el número de bytes que faltan para completar el tamaño del bloque siguiente.

Es decir:
Si: Input (ASCII): H O L A _ _ _ _ _ _ _ _ _ _ _ _
Entonces: Input + relleno 0x: 48 4f 4c 41 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c

Como vemos, la cadena HOLA tiene 4 bytes, con lo cual faltan 12 bytes más para llegar a 16 bytes que es el tamaño del bloque de entrada, 128 bits. Como faltan 12 bytes y el valor 12 en hexadecimal es 0c, entonces se rellena con 0c los 12 bytes que faltan del bloque.

Incluso si la entrada fuese un múltiplo del tamaño de bloque, se rellenaría con el valor 10 en hexadecimal (correspondiente al 16 decimal, igual al tamaño de un bloque) que es lo que ha sucedido en el ejercicio del aparatado anterior.

Con la clave de 128 bits en modo ASCII K que se indica, cifra el siguiente mensaje M en modo ECB y observa el relleno.

M = Vamos a ver qué pasa ahora con el relleno K: MiGranClaveMiedo
¿De qué tamaño es el relleno y porqué tiene ese valor?
¿En qué formato se encuentra la salida por defecto?
¿Por qué se usa ese formato especial?

X CIFRADO SIMÉTRICO EN BLOQUES -AES. Cifrado y descifrado en pantalla en modo ECB

10.1 Con modo de ejecución directo, cifra el texto M con entrada en pantalla usando la clave de 128 bits, en hexadecimal que se indicada y con la salida en hexadecimal. Usa la opción de copiar que verás en Editar del menú del programa, o bien copiar y pegar con las teclas Control C y Control V.

Mensaje M

De hecho, el AES, a pesar de ser un algoritmo público y de uso público, está considerado como la NSA (Agencia de Seguridad Nacional de los EEUU) desde el año 2003, como algoritmo seguro para proteger información clasificada de nivel SECRETO usando claves de 128 bits y de ALTO SECRETO, si se usan claves de 192 o 256 bits. No deja de sorprender, teniendo en cuenta el amor-odio del Gobierno de los Estados Unidos con los sistemas de cifrado, que el gran público pueda tener acceso a un sistema de cifrado considerado apto por la NSA, para proteger información sensible del más alto nivel, lo que no deja de levantar muchas suspicacias.

Fuente: parte del documento ¿Fue buena idea usar AES256 con el archivo INSURANCE de Wikileaks? de Fernando Acero, http://fernando-acero.livejournal.com/78069.html

Clave K1 = **Aquí16caracteres**
Copia el criptograma y descífralo.
Repite el ejercicio anterior con las claves ahora en hexadecimal K2 de 192 bits y K3 de 256 bits. En estos casos la salida será en código Base64.
K2 = **4142434445464748494a4b4c4d4ed14f5051525354555657**
K3 = **6162636465666768696a6b6c6d6ef16f707172737475767778797a3132333435**

10.2 Cifrado de un archivo y descifrado con archivos en modo ECB Busca en tu ordenador algún archivo en Word y cífralo con la siguiente clave de 128 bits en hexadecimal K1 = **0123456789ABCDEFFEDCBA9876543210**.

La salida será el archivo .**ESphere**. Hecho esto, descifra el .**Sphere** siendo la salida en un archivo. Abre la carpeta y ponle a este archivo la extensión .docx. Ábrelo y comprueba que obtienes el mismo archivo original.

XI CIFRADO SIMÉTRICO EN BLOQUES -AES. Cifrado y descifrado en pantalla en modo CBC

11.1 Cifra el siguiente texto M en modo CBC siendo las claves de 128 bits K e IV, salida en hexadecimal.
K = **0123456789ABCDEF** = **0x 30313233343536373839414243444546**
IV 0x: **00000000000000000000000000000000**
M = **ABCD**

11.2 M = **414243440c0c0c0c0c0c0c0c0c0c0c0c** C = **98b50d9f6a96299434fbdf8fcd4b1754**

11.3 ¿Se diferencia de la cifra ECB? ¿Por qué? ¿Y si el vector de inicialización IV fuese 0x: 00000000000000000000000000000001?

11.4 Cifrar ahora el siguiente texto M en modo CBC siendo las claves de 128 bits K e IV, salida en Base64.

K = 0123456789ABCDEF = 0x 30313233343536373839414243444546

IV = 123VectorInicial = 0x 313233566563746f72496e696369616c

M = Una cifra con AES en modo Cipher Block Chaining CBC.

Descifrar el criptograma y recupera el texto en claro.

11.5 Buscar un archivo de cualquier tamaño y cífralo en modo CBC con las claves K1 de 128 bits y K2 de 192 indicadas en ASCII, con un vector de inicialización
IV 0x: **00000000000000000000000000000001**:
K1 = **abcdefghijklmnñopqrstuvw**

K2 = **EstaMateriaSeLlamaCriptografiaAplicada**

11.5.1 Si las claves K1 y K2 tienen tamaños distintos, ¿por qué los archivos cifrados tienen igual tamaño?

11.5.2 ¿El tamaño del archivo cifrado es mayor o menor que el archivo en claro? ¿Por qué?

11.6 Cifrar con el modo de ejecución Paso a Paso el siguiente texto en modo ECB y CBC con las claves que quieras de 128, 192 y 256 bits y observa lo que te entrega la pantalla de salida.

M = En este momento, el huracán Sandy se desplaza a una velocidad de 44 kilómetros por hora y se encuentra a menos de 200 kilómetros al sureste de Atlantic City (Nueva Jersey) y a 285 kilómetros de la ciudad de Nueva York. Sigue avanzando con vientos sostenidos de más de 145 kilómetros por hora, según un comunicado del Centro Nacional de Huracanes.

11.6.1 ¿Cuántos bloques debe cifrar?

11.6.2 Hacer lo mismo con un archivo cualquiera de al menos 100 KB y observa cuántos bloques ha cifrado el algoritmo.

XII CIFRADO SIMÉTRICO EN BLOQUES - Ataque monousuario cliente - servidor

Si el texto cifrado en Base64 y el texto en claro en hexadecimal son los que se indican, se pide realizar el ataque en formato monousuario dentro del siguiente espacio de claves en ASCII de 128 bits:
Clave inicial ASCII: **MiSuperClave1111**
Clave final ASCII: **MiSuperClave1333**
M = 0x
56616d6f732061207665722063f36d6f2066756e63696f6e61206573746f2064656c20617461717565
20656e206d6f646f206c6f63616c2e0808080808080808
C=B64
syJc3+VyOarMjWvDi2xTrMvTCGfZenXvgeXSNiWulbQVFmeBIchbMha0LmxdwQV1d5PaZ7D4iJt
BIMgVNoAhiQ==

Cuando el programa encuentre la clave de cifra, se pide descifrar el criptograma y comprobar que el texto se corresponde a la cadena hexadecimal utilizada como texto en claro durante el ataque. ¿A qué texto en ASCII corresponde el mensaje?