Tarea 2: Criptograma de llave simétrica.

Resumen:

En este trabajo se retomará la tarea 1 para realizar un esquema criptográfico. Se utilizará Python para la obtención de parámetros, la encriptación y desencriptación por bloques de 16 bits, para esto fue necesario usar los códigos de Huffman. Se retomaron conceptos y funcionamiento de la encriptación RSA.

Introducción:

Se reportarán resultados de diversos mensajes encriptados de distintas longitudes, así como diversas combinaciones en las llaves.

Descripción del problema:

Se necesitaba desarrollar un script de Python, que utilizando la tarea 1 sobre códigos de Huffman, lograra encriptar y desencriptar un mensaje. Dicho script utilizó un procedimiento parecido a RSA. El programa obtiene del usuario 2 números primos **p** y **q**, que multiplicados dan lugar a **n** para nuestro campo finito $\mathbf{F_n}$. Obtenemos el **mcm(p-1, q-1)** de estos números, y nuestro parametro **e** debe ser menor que este resultado y a su vez ser coprimo. Una vez seleccionado estos parametros se procede a calcular el inverso multiplicativo de **e** respecto al campo finito $\mathbf{F_n}$, dando lugar a nuestro tercer parametro **d** para la llave secreta. Usando exponenciación rápida se obtiene el cifrado para cada bloque de 16 bits con $\mathbf{P^e \bmod n}$. De igual manera, se obtiene la decodificación con $\mathbf{C^d \bmod n}$. Dada la codificación de Huffman se obtuvo el número en decimal para la exponenciación en la codificación, y después de la decodificación se obtuvieron los valores de decimal a binario para su respectiva traducción con Huffman.

Descripción del programa desarrollado en Python:

Se hace un llamado a la tarea 1, para traer los métodos de codificación y decodficación Huffman(***import*** *HuffmanEntropy*).

Obtención de parámetros: Parametros()

Simple y sencillamente se dan por consola los números **p**, **q**, y **e** ya mencionados, y se procede a obtener el inverso modular $\mathbf{d = e^{-1} \bmod n}$ con el algoritmo de Euclides extendido. Si **e** no cumple con sus condiciones, se vuelve a pedir.

Encriptación: Encrypt(Mensaje, Llave)

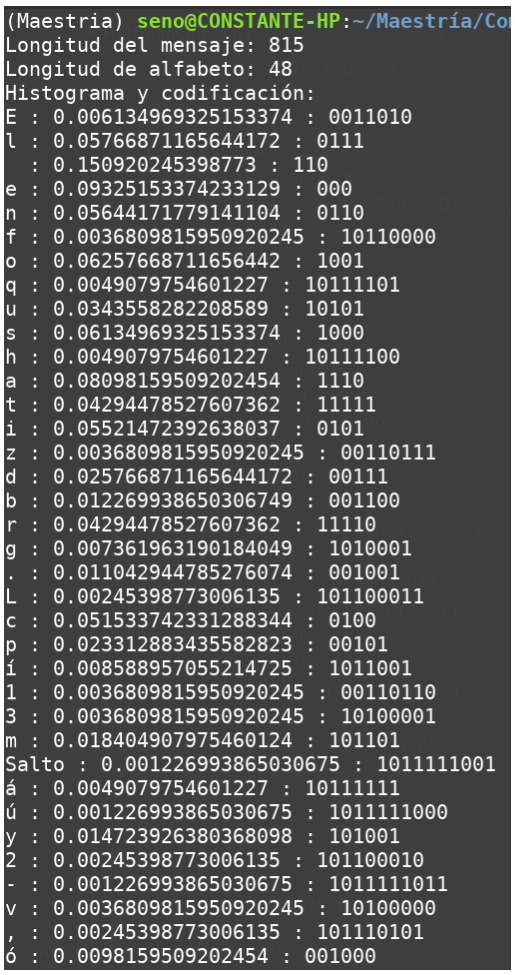
Se procede a codificar en Huffman el mensaje inicialmente dado, y se preparan los bloques de 16 bits para proceder encriptar con $\mathbf{P^e \bmod n}$ cada bloque, se utilizó la transformación de binario a decimal y la exponenciación rápida para campos finitos.

Desencriptación: Decrypt(Mensaje cifrado)

Se procede a pedirle al usuario por consola su llave secreta, para desencriptar dicho mensaje. Se utiliza la exponenciación rápida para desencriptar $\mathbf{C^d \bmod n}$, se transforman los bloques resultantes de decimal a binario y se traduce con códigos de Huffman, obteniendo así en mensaje inicial.

Resultados:

Se hace conocer el diccionario usado para la codificación**:**



Se procede a dar los parametros y el mensaje a encriptar(p en 16 bits):

Ingresar mensaje*: A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message*

Ingresa p: *727*

Ingresa q*: 787*

*95106.0 # Resultado de q*p*

Ingresa e*: 103*

Valores*: n = 572149 e = 103 d = 59095.0 # d calculada como el inverso multiplicativo de e*

Llave secreta: *(572149, 103, 59095.0) #Llave secreta generada*

Mensaje y su codificación en Huffman:

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message : ['111100', '1110101011100001', '1110001110110110', '1011000110010111', '1100010111110001', '1010111001101010', '1100111110100011', '1011111000001111', '1011101010100101', '1101111101110010', '1100001011011111', '1111100000100001', '1100100011111100', '1101000011110111', '1011011111111100', '1000100001100100', '1001110111100111', '1100001000101010', '1010001101011100', '1000101101100101', '1011011111011100', '1110001001101011', '1011011110110000', '1000001011111111', '1110101001100100',

'1001010101110001', '1100100010000001', '1110101111000110', '1101011101011010',
'1010101100000010', '1111100110000100', '1111101110010110', '1111010101101110',
'1111000111001101', '1101000110010011', '1011110110001010', '1110000111110101',
'1101100000011111', '1011011100111001', '1110010100010111', '1111111101010011',
'1001000010101011', '1100011001000100', '1000011101011110', '1111001011000011',
'1111110101001000', '1000101111001111', '1000011111111110', '1101011100000111',
'1110100011110111', '1110110111100110', '1100101100010100', '1101010001010011',
'1110101011011100', '1001110001011111', '1011100101000101', '1111111110000010',
'1000110010000111', '1011110011110000', '1100010101111101', '1001101000010100',
'1111110110001011', '1111111110101110', '1110111001000001', '1110111110110000',
'1000001000110101', '1010010001100011', '1110101100101010', '1110100010101011',
'1110011100100001', '1001010100010111', '1100110000100111', '1101110010110111',
'1100111100101100', '1111100110000110', '1000001011000000', '1000101100110001',
'1100101111101110', '1010100010111111', '1111101010011001', '1000010101011100',
'1011001000100000', '1011101011110100', '1111101010110100', '1101000001110010',
'1010100010111110', '1111001010001011', '1100100000111011', '1111011000000000']

Codificación Huffman en decimal:

[60, 60129, 58294, 45463, 50673, 44650, 53155, 48655, 47781, 57202, 49887, 63521, 51452, 53495, 47100, 34916, 40423, 49706, 41820, 35685, 47068, 57963, 47024, 33535, 60004, 38257, 51329, 60358, 55130, 43778, 63876, 64406, 62830, 61901, 53651, 48522, 57845, 55327, 46905, 58647, 65363, 37035, 50756, 34654, 62147, 64840, 35791, 34814, 55047, 59639, 60902, 51988, 54355, 60124, 40031, 47429, 65410, 35975, 48368, 50557, 39444, 64907, 65454, 60993, 61360, 33333, 42083, 60202, 59563, 59169, 38167, 52263, 56503, 53036, 63878, 33472, 35633, 52206, 43199, 64153, 34140, 45600, 47860, 64180, 53362, 43198, 62091, 51259, 62976]

Encriptación decimal:

[60, 60129, 58294, 45463, 50673, 44650, 53155, 48655, 47781, 57202, 49887, 63521, 51452, 53495, 47100, 34916, 40423, 49706, 41820, 35685, 47068, 57963, 47024, 33535, 60004, 38257, 51329, 60358, 55130, 43778, 63876, 64406, 62830, 61901, 53651, 48522, 57845, 55327, 46905, 58647, 65363, 37035, 50756, 34654, 62147, 64840, 35791, 34814, 55047, 59639, 60902, 51988, 54355, 60124, 40031, 47429, 65410, 35975, 48368, 50557, 39444, 64907, 65454, 60993, 61360, 33333, 42083, 60202, 59563, 59169, 38167, 52263, 56503, 53036, 63878, 33472, 35633, 52206, 43199, 64153, 34140, 45600, 47860, 64180, 53362, 43198, 62091, 51259, 62976]

Obtención de llave por consola:

Ingresar datos de llave:
Ingresa n: 572149
Ingresa e: 103
Ingresa d: 59095
 # Si la llave coincide, la desencriptación es correcta

Desencriptación en códigos de Huffman:

['11100', '110101011100001', '110001110110110', '011000110010111', '100010111110001', '010111001101010', '100111110100011', '011111000001111', '011101010100101', '101111101110010', '100001011011111', '111100000100001', '100100011111100', '101000011110111', '011011111111100', '000100001100100', '001110111100111', '100001000101010', '010001101011100', '000101101100101', '011011111011100', '110001001101011', '011011110110000', '000001011111111', '110101001100100', '001010101110001', '100100010000001', '110101111000110', '101011101011010', '010101100000010', '111100110000100', '111101110010110', '111010101101110', '111000111001101', '101000110010011', '011101110001010', '000011101100111', '110001000101010', '010001101011100', '000101101100101', '011011111011100', '110001001101011', '011011110110000', '000001011111111', '110101001100100', '001010101110001', '100100010000001', '110101111000110', '101011101011010', '010101100000010', '111100110000100', '111101110010110', '111010101101110', '111000111001101', '101000110010011', '011110110001010', '110000111110101', '101100000011111', '011011100111001', '110010100010111', '111111101010011', '001000010101011', '100011001000100', '000011101011110', '111001011000011',

'111110101001000', '000101111001111', '000011111111110', '101011100000111', '110100011110111', '110110111100110', '100101100010100', '101010001010011', '110101011011100', '001110001011111', '011100101000101', '111111110000010', '000110010000111', '011110011110000', '100010101111101', '001101000010100', '111110110001011', '111111110101110', '110111001000001', '110111110110000', '000001000110101', '010010001100011', '110101100101010', '110100010101011', '110011100100001', '001010100010111', '100110000100111', '101110010110111', '100111100101100', '111100110000110', '000001011000000', '000101100110001', '100101111101110', '010100010111111', '111101010011001', '000010101011100', '011001000100000', '011101011110100', '111101010110100', '101000001110010', '010100010111110', '111001010001011', '100100000111011', '111011000000000']

Mensaje traducido:

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message

Conclusiones:

Para comenzar en la criptografía se debe conocer los fundamentos más básicos de un esquema de encriptación de datos, esta vez se puso en práctica los conocimientos obtenidos de algebra modular. S implemento un sistema de encriptación por bloques el cual funcionó correctamente con la teoría de campos finitos. Se investigó como se obtienen parametros adecuados para la exponenciación modular e incrementar la seguridad del cifrado.

Referencias:

Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" . Communications of the ACM. 21 (2): 120–126. CiteSeerX 10.1.1.607.2677. doi:10.1145/359340.359342