TAREA02
CRIPTOGRAFÍA Y SEGURIDAD

Ejercicio 1:

a) 2^56 = 72,057,594,037,927,936?
10^10 = 10,000,000,000 claves en un segundo
=? 600,000,000,000? claves en un minuto
= 36,000,000,000,000 claves en una hora
= 864,000,000,000,000? claves en un dia
72,057,594,037,927,936 / 864,000,000,000,000
Respuesta; 83.399 dias

b) 864,000,000,000,000? claves en un dia
31,557,600,000,000,000,000? claves en un año
3,155,760,000,000,000,000,000 claves en 100 años
un aproximado seria 2^71.419 que es 3,156,903,163,540,658,027,821.5238064277? claves
pero no usando decimales en la potencia seria 2^72 igual a 4,722,366,482,869,645,213,696? claves pero serian 149.64 años.

Ejercicio 2: ¿Los siguientes esquemas de cifrado son perfectamente seguros? Explica.
a) El esquema de cifrado es considerado perfectamente seguro si cumple que cada llave es elegida con igual probabilidad, es decir 1/k, por el algoritmo KeyGen y justo como se toma una clave al azar del conjunto K, se cumple que su probabilidad es 1/k.
También debe cumplirse para que sea perfectamente seguro que para cada mensaje y cada texto cifrado, exista una única llave k tal que Enc(k,m)=c, y sabemos que Enck(m)cumple que tiene un único c porque cuando se elige una clave y se le suma cada mensaje claro, se obtienen números consecutivos que al aplicarel módulo solo un c para cada mensaje m.
Entonces si es perfectamente seguro el esquema de cifrado.

b)Sabemos que hay un solo c ya que la probabilidad al elegir una llave de entre las 27 posibles es 1/k también, entonces siempre será el mismo desplazamiento de los k lugares.
Entonces si es perfectamente seguro el esquema de cifrado.

c)Para cifrar sabemos que éste esquema genera una secuencia de claves verdaderamente aleatorias y luego se combina con el texto plano por una operación XOR. La operación binaria XOR se realiza en cada bit con el bit correspondiente en la clave secreta entonces obliga a que la llave sea de la longitud del mensaje, como la llave es generada aleatoriamente, c se hace único y con ello, perfectamente seguro.

Ejercicio 3:

El adversario genera dos mensajes m0,m1 pertenece M
Escoje un bit al azar ya sea 1 o 0 y aplicar one time pad ya que ocupa XOR
el adebersario recibe el mensaje cifrado pero dado que 1 y 0 plicando XOR nos dara el mismo resultado entonces siempre ganara el adbersario

Ejercicio 5:

Lo decodifique en python pero alquere copiar la la decodificacion se trababa y no podia copear.
¿Cual ocupa mayor espacio? Los dos ocupan el mismo espacio

¿Porque? por que el exadecimal es Base16 y tanto base16 como base64 convierten bytes de 8 bits a valores que caben dentro del rango de caracteres imprimibles ASCII.

Ejercicio 6: Para cifrar un mensaje usando One-time pad necesitamos una cadena aleatoria de bytes. Supongamos que Bartolo quiere enviar un archivo a Alicia, entonces Juan Aleatorio le proporciona el archivo cinta aleatoria.txt, que contiene una cadena de bytes k en Base64. Bartolo usa k como llave para cifrar el archivo imagen.png y envía el resultado c a Alicia.
Alicia, que también posee k, al recibir el mensaje cifrado c revisa los primeros bytes y nota algo extraño: c no parece algo aleatorio, sino que es un mensaje de un formato muy particular.
Cuando Alicia descifra c obtiene el mensaje correcto (el archivo imagen.png), y entonces se da cuenta de que la llave no es una cadena tan arbitraria de bytes, al parecer Juan Aleatorio conocía el mensaje que Bartolo quería enviar.

a) Para cifrar imagen.png se uso el programa que esta en la carpeta llamado "converter.py", se reviso que tuvieran la misma longitud que el mensaje y efectivamente los dos eran de longitud 400663, y tal vez Alicia noto algo extraño porque sus primeros dos bytes del archivo ya cifrado valen 255 y 251 que en hexadecimal son el ff y el fb y asi inician las imagenes en formato png.

b)Suponiendo que k fue aleatorio entonces no sabemos mucho mas que lo evidente entonces la probabilidad de que m + k sea el c que recibió Alicia se ve como
Pc(m)= P(m), es decir 1/k por cada byte.

c)Solo se me ocurre que pudo haber tomado el archivo imagen.png ya que parecia que si lo conocía y le aplico su negacion a toda su cadena de bytes.