Российский университет дружбы народов Научный факультет

Математические основы защиты информации и

информационной безопасности

шифрование de gamma

Подготовлено студентом:

Елиенис Санчес Родригес.

Преподаватель: Дмитрий Сергеевич

Cifrado de gamma

el método simétrico de cifrado, que consiste en la "imposición" de la secuencia formada por números aleatorios, en el texto. La secuencia de números aleatorios se llama gamma de la secuencia y se utiliza para зашифровывания y descifrar datos. La suma se realiza generalmente en alguno de los el campo de destino.

El texto cifrado obtenido por este método es bastante difícil de descifrar, ya que la clave aquí es variable. Esencialmente, la gamma de cifrado debe cambiar aleatoriamente para que se cifre cada bloque. Si el período gamma supera la longitud de todo el texto cifrado y el atacante no conoce ninguna parte del texto original, dicho cifrado solo puede resolverse mediante la enumeración directa de todas las opciones clave. En este caso, la fuerza del cifrado está determinada por la longitud de la clave.

Sin embargo, el método gamma se vuelve impotente si el atacante reconoce un fragmento del texto fuente y el cifrado correspondiente. Mediante una simple resta de módulo, se obtiene un segmento de una secuencia pseudoaleatoria y se restaura toda la secuencia a partir de él.

 El cifrado gamma utiliza una cadena aleatoria de bits como clave y la combina con el texto sin formato, también representado en binario, mediante la adición bit a bit módulo 2 para producir texto cifrado. La generación de secuencias binarias impredecibles de gran longitud es uno de los problemas importantes de la criptografía clásica. Para resolver este problema, los generadores de secuencias pseudoaleatorias binarias son ampliamente utilizados.

La serie de números pseudoaleatorios generada a menudo se denomina gamma de cifrado o simplemente gamma (por el nombre de la letra griega ***g***   , que a menudo se usa en fórmulas matemáticas para denotar variables aleatorias).

Por lo general, para generar una secuencia de números pseudoaleatorios, se utilizan programas informáticos que, aunque se denominan generadores de números aleatorios, en realidad generan secuencias de números deterministas que son muy similares en sus propiedades a las aleatorias.

Hay tres requisitos principales para un generador de números PSP criptográficamente seguro (cifrado gamma):

el período gamma debe ser lo suficientemente grande como para cifrar mensajes de varias longitudes;

la gamma debe ser prácticamente impredecible, lo que significa que es imposible predecir el siguiente bit de la gamma, incluso si se conocen el tipo de generador y el trozo de gamma anterior;

la generación gamma no debería causar grandes dificultades técnicas;

La duración del período gamma es la característica más importante del generador FP. Al final del período, los números comenzarán a repetirse y se podrán predecir.

Una de las primeras formas de generar un PSCH en una computadora fue propuesta en 1946 por John von Neumann. La esencia de este método es que cada número aleatorio posterior se forma elevando al cuadrado el número anterior, descartando los dígitos inferior y superior. Sin embargo, este método resultó poco fiable y pronto fue abandonado.

De los procedimientos conocidos para generar una secuencia PRNG, el llamado *generador lineal congruente* es el más utilizado . Este generador genera una secuencia PRNG Y 1 , Y 2 , …, Y i-1 , Y i , … utilizando la relación

**Yi = (a • Yi -1 + b) módulo m,**

donde Y i         es el i-ésimo número (actual) de la secuencia;

Y i-1       número de secuencia anterior;

m es el módulo;

a es un multiplicador;

b - incremento;

y Y 0 es el número generador (valor inicial).

El número pseudoaleatorio actual Y i se obtiene del número anterior Y i-1 multiplicándolo por un factor a, sumando con un incremento de b y calculando el resto después de dividir por m. Esta ecuación genera un PRNG con un periodo de repetición en función de los valores elegidos de a y b y puede llegar al valor m. El valor de m generalmente se establece en 2 n , donde n es la longitud de la palabra de la máquina en bits, o en un número primo, como m=2 31 -1. Como lo muestra D. Knuth, un sensor PRNG congruente lineal tiene un período máximo si y solo si b es impar y mod 4 = 1.

Además, se utilizan generadores aditivos y multiplicativos para obtener una secuencia PRNG.

*El generador multiplicativo* genera secuencias de números usando la relación de recurrencia:

**Y i = (a • Y i-1 ) mod m.**

Los requisitos para los valores de las constantes a y m son los mismos que para el generador lineal congruente.

El número aleatorio actual Y i del*sensor aditivo* se obtiene de la suma de los números Y i-1 e Y i-2 calculando el módulo de dividir esta suma por m:

**Yi = (Yi -1 + Yi -2 ) mod m.**

El algoritmo de conversión es el siguiente: inicializar el sensor PSC (1); seleccione un bloque de texto sin formato; generar GSH (2); obtenga el bloque de texto cifrado mediante el módulo 2 agregando el bloque de texto sin formato al GSh (3); si el texto no está terminado, pase al punto 2, de lo contrario pase al punto 5 (4); fin del algoritmo de cifrado (5).

El algoritmo de transformación inversa se realiza de la siguiente manera: inicialice el sensor PSC; seleccione el bloque de texto cifrado (1); generar GSH (2); Obtener un bloque de texto plano por módulo 2 agregando un bloque de texto cifrado con GSH (3); si el texto cifrado no está terminado, vaya al paso 2, de lo contrario, vaya al paso 6 (4); final del algoritmo de descifrado (5).

from pyfiglet import figlet\_format  
print(figlet\_format( "Cifrado de gama by Elienis", font = "cybermedium"))  
archivo = open('Sourse.txt','w')  
n= 0  
while n < 1:  
 texto=input('ingresa la frase a encriptar' )  
 archivo.write(texto+'\n')  
 n=n+1  
  
archivo.close()  
  
A = 15  
B = 17  
M = 4096  
Y0 = 4003  
  
  
def Gamma(y):  
 gamma\_list = []  
 for \_ in range(8):  
 y = (A \* y + B) % M  
 gamma\_list.append(y)  
 return gamma\_list  
  
  
def Crypt():  
 gamma = Gamma(Y0)  
 res = open("Result.txt", "w", encoding="utf-8")  
 with open('Sourse.txt', 'r', encoding="utf-8") as f:  
 r\_int = ""  
 r = ""  
 while True:  
 temp = f.read(8)  
 if temp:  
 for i, item in enumerate(temp):  
 r\_int = r\_int + " " + str(ord(item) ^ gamma[i])  
 r = r + " " + chr(ord(item) ^ gamma[i])  
 res.write(chr(ord(item) ^ gamma[i]))  
  
 else:  
 break  
 print(r\_int)  
 res.close()  
  
  
Crypt()  
  
  
def DeCrypt():  
 gamma = Gamma(Y0)  
 res = open("NewResult.txt", "w", encoding="utf-8")  
 with open('Result.txt', 'r', encoding="utf-8") as f:  
 r\_int = ""  
 r = ""  
 while True:  
 temp = f.read(8)  
 if temp:  
 for i, item in enumerate(temp):  
 r\_int = r\_int + " " + str(ord(item) ^ gamma[i])  
 r = r + chr(ord(item) ^ gamma[i])  
 res.write(chr(ord(item) ^ gamma[i]))  
 else:  
 break  
 print(r\_int)  
 res.close()  
  
  
DeCrypt()  
  
print("a continuacion se mostrara el texto cifrado\n")  
with open("C:/Users/kami/Documents/matematica/lab03/Result.txt","r") as archivo:  
 for linea in archivo:  
 print(linea)



