Российский университет дружбы народов Научный факультет

Математические основы защиты информации и

информационной безопасности

Гамма-шифрование

Подготовлено студентом:

Елиенис Санчес Родригес.

Преподаватель: Дмитрий Сергеевич

Гамма-шифрование

симметричный метод шифрования, заключающийся в "наложении" последовательности, образованной случайными числами, на текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровки данных. Суммирование обычно выполняется в одном из целевое поле.

Зашифрованный текст, полученный этим методом, довольно сложно расшифровать, поскольку ключ здесь является переменным. По сути, гамма шифрования должна изменяться случайным образом, чтобы каждый блок был зашифрован. Если период гамма превышает длину всего зашифрованного текста и злоумышленник не знает ни одной части исходного текста, такое шифрование может быть разрешено только путем прямого перечисления всех ключевых параметров. В этом случае надежность шифрования определяется длиной ключа.

Однако гамма-метод становится бессильным, если злоумышленник распознает фрагмент исходного текста и соответствующий шифр. Простым вычитанием по модулю получается отрезок псевдослучайной последовательности и из него восстанавливается вся последовательность.

Гамма-шифрование использует случайную строку битов в качестве ключа и объединяет ее с обычным текстом, также представленным в двоичном формате, путем побитового добавления по модулю 2 для получения зашифрованного текста. Генерация непредсказуемых двоичных последовательностей большой длины - одна из важных проблем классической криптографии. Для решения этой проблемы широко используются бинарные генераторы псевдослучайных последовательностей.

Сгенерированный псевдослучайный числовой ряд часто называют шифрующей гаммой или просто гаммой (по названию греческой буквы***g*** , которая часто используется в математических формулах для обозначения случайных величин).

Обычно для генерации последовательности псевдослучайных чисел используются компьютерные программы, которые, хотя и называются генераторами случайных чисел, на самом деле генерируют последовательности детерминированных чисел, которые по своим свойствам очень похожи на случайные.

Существует три основных требования к криптографически безопасному генератору чисел PSP (гамма-шифрование):

период гамма должен быть достаточно большим для шифрования сообщений различной длины;

гамма должна быть практически непредсказуемой, что означает, что невозможно предсказать следующий бит гаммы, даже если тип генератора и предыдущий фрагмент гаммы известны;

гамма-генерация не должна вызывать больших технических трудностей;

Продолжительность гамма-периода является наиболее важной характеристикой генератора FP. В конце периода числа начнут повторяться, и их можно будет предсказать.

Один из первых способов создания PSCH на компьютере был предложен в 1946 году Джоном vonфон Нейманом. Суть этого метода в том, что каждое последующее случайное число формируется путем возведения в квадрат предыдущего числа с отбрасыванием нижней и верхней цифр. Однако этот метод оказался ненадежным и вскоре от него отказались.

Из известных процедур генерации последовательности*ГСЧП наиболее часто используется так называемый конгруэнтный линейный генератор* .Этот генератор генерирует последовательность PRNG И1, И 2,..., Иi-1, Иi, ... используя соотношение

**Yi = (а • Yi-1 +b) по модулю m,**

где Иi- i-ésimoе (текущее) число в последовательности;

Иi-1предыдущий порядковый номер;

m-модуль;

а-множитель;

б - приращение;

и И0-это число генератора (начальное значение).

Текущее псевдослучайное число Иiполучается из предыдущего числа Иi-1, умножая его на множитель а, суммируя с приращением b и вычисляя остаток после деления на m. Это уравнение генерирует ГСЧ с периодом повторения на основе выбранных значений а и в и может получить при значении m. Значение m обычно устанавливается равным 2n, где n - длина машинного слова в битах или простое число, например m=2 31-1. Как показывает Д. Кнут, линейный конгруэнтный датчик ГСЧП имеет максимальный период тогда и только тогда если b нечетно, а mod 4 = 1.

Кроме того, для получения последовательности PRNG используются аддитивные и мультипликативные генераторы.

*Мультипликативный генератор*генерирует последовательности чисел с использованием рекуррентного отношения:

**Y i = (a • Y i-1 ) mod m.**

Требования к значениям констант а и м такие же, как и к конгруэнтному линейному генератору.

Текущее случайное число Иi*аддитивного датчика*получается из суммы чисел Иi-1и иi-2путем вычисления модуля деления этой суммы на m:

**Yi = (Yi -1 + Yi -2 ) mod m.**

Алгоритм преобразования следующий: инициализировать датчик PSC (1); выберите блок открытого текста; сгенерировать GSH (2); получить блок зашифрованного текста с помощью модуля 2, добавив блок открытого текста в GSh (3); если текст не завершен, перейдите к пункту 2, в противном случае перейдите к пункту 5 (4); конец алгоритма шифрования (5).

Алгоритм обратного преобразования выполняется следующим образом: инициализируем датчик PSC; выберите блок зашифрованного текста (1); сгенерировать GSH (2); Получить блок открытого текста по модулю 2, добавив блок зашифрованного текста с помощью GSH (3); если зашифрованный текст отсутствует, выполните следующие действия. готово, перейдите к шагу 2, в противном случае перейдите к шагу 6 (4); завершение алгоритма дешифрования (5).

from pyfiglet import figlet\_format  
  
#importamos la funcion para colocar letras en la portada  
#импортируем функцию размещения букв на обложке  
  
print(figlet\_format( "Cifrado de gama by Elienis", font = "cybermedium"))  
  
#crearemos un archivo que se llama Source.txt y comenzaremos a escrivir en el  
#мы создадим файл Source.txt и начнем в него писать  
  
archivo = open('Sourse.txt','w')  
n= 0  
while n < 1:  
 texto=input('ingresa la frase a encriptar' )  
 # вводим в файл фразу для шифрования  
 #introducimos en el archivo la frase a encriptar  
 archivo.write(texto+'\n')  
 n=n+1  
  
archivo.close()  
#cerramos el archivo  
# закрыть файл  
  
# cadena aleatoria de bits como clave y la combina con el texto sin formato  
# случайная строка битов в качестве ключа и комбинирует ее с обычным текстом  
A = 15  
B = 17  
M = 4096  
Y0 = 4003  
  
#definimos gamma y denotamos las variables aleatorias  
#определить гамму и обозначить случайные величины  
def Gamma(y):  
 gamma\_list = []  
 for \_ in range(8):  
 y = (A \* y + B) % M  
 gamma\_list.append(y)  
 return gamma\_list  
  
#definimos la funcion encriptar, abrimos el txt de entrada Source y escribimos en Result  
#определить функцию шифрования, открыть исходный входной txt и записать в результат  
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])  
 # объединить гамма-переменную с введенным текстом  
 #combinamos la variable gama con el texto intriducido  
 r = r + " " + chr(ord(item) ^ gamma[i])  
 res.write(chr(ord(item) ^ gamma[i]))  
  
 else:  
 break  
 print(r\_int)  
 res.close()  
#guardamos en resultado en el archivo Result.txt  
# сохранить результат в файле Result.txt  
Crypt()  
  
# definimos la funcion para desencriptar y generar un archivo llamado NewResult  
# определить функцию для расшифровки и создания файла с именем NewResult  
def DeCrypt():  
 gamma = Gamma(Y0)  
 #crearemos un nuevo archivo que contendra la frase decifrada  
 # создаст новый файл, который будет содержать расшифрованную фразу  
 res = open("NewResult.txt", "w", encoding="utf-8")  
  
 # abrimos el archivo con la frase cifrada  
 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):  
  
 # realizamos el decifrado de gamma  
 # выполнить гамма-расшифровку  
  
 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)  
 #imprimimos el resultado y lo grabamos  
 # распечатываем результат и сохраняем  
 res.close()  
  
  
DeCrypt()  
  
print("a continuacion se mostrara el texto cifrado\n") # распечатать результаты шифрования  
with open("C:/Users/kami/Documents/matematica/lab03/Result.txt","r") as archivo:  
 for linea in archivo:  
 print(linea)  
  
print("a continuacion se mostrara el texto decifrado\n") # вывести расшифрованные результаты  
with open("C:/Users/kami/Documents/matematica/lab03/NewResult.txt","r") as archivo:  
 for linea in archivo:  
 print(linea)



