МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №1

З дисципліни «Технології захисту інформації»

Виконав: *студент групи КН-310*
Бікєєв Андрій

Викладач:
Яковина В. С.

Львів – 2020

Варіант 2

Тема: Створення генератора псевдовипадкових чисел.

Мета: Ознайомитись з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел та навчитись створювати програмні генератори псевдовипадкових чисел для використання в системах захисту інформації.

Теоретичні відомості

Сучасна інформатика широко використовує випадкові числа в різних програмах – від методу Монте-Карло до криптографії. Ряд алгоритмів захисту мережі, заснованих на засобах криптографії, передбачає використання випадкових чисел. Ці застосування висувають дві вимоги до послідовності випадкових чисел: випадковість і непередбачуваність

Джерелами дійсно випадкових чисел потенційно можуть бути фізичні генератори шумів, такі як імпульсні детектори іонізуючого випромінювання, газорозрядні лампи, конденсатори з втратами струму тощо. Однак такі пристрої можуть знайти доволі обмежене застосування в додатках для захисту інформації. Туту існують проблеми як з випадковістю, так і з точністю отриманих таким методом чисел, не кажучи вже про проблеми підключення такого роду пристроїв до кожної системи в мережі.

Тому криптографічні додатки зазвичай використовують алгоритмічні методи генерування випадкових чисел. Відповідні алгоритми є детермінованими і тому породжують послідовності чисел, які не є статистично випадковими. Однак, якщо алгоритм є достатньо хорошим, породжувані ним послідовності чисел витримують багато тестів на випадковість. Такі числа часто називають псевдовипадковими.

Генератор псевдовипадкових чисел – алгоритм, що генерує послідовність чисел, елементи якої незалежні один від одного і підлягають заданому розподілу.

Найбільш популярним алгоритмом для генерування псевдовипадкових чисел є алгоритм, запропонований Лемером, який називається методом лінійного порівняння. Цей алгоритм має чотири наступних параметри.

m модуль порівняння $m > 0$

a множник $0 \le a < m$

c приріст $0 \le c < m$

X_0 початкове число $0 \le X_0 < m$

Послідовність псевдовипадкових чисел {X0} отримують за допомогою ітерацій наступного співвідношення:

$$X_{n+1} = (aX_n + c) \bmod m$$

При цьому якщо m, a, c та X0 є цілими, то буде отримано послідовність цілих чисел з діапазону $0 \le X_n < m$. Вибір значень для a, c та m є дуже важливим з точки зору створення хорошого генератора псевдовипадкових чисел.

Пропонується три критерії, за якими можна оцінити якість будь-якого генератора псевдовипадкових чисел.

1. Функція генерації повинна бути функцією повного періоду, тобто функція повинна породити усі числа від 0 до m перед тим, як числа почнуть повторюватись.
2. Створена послідовність повинна вести себе як випадкова. Насправді ця послідовність не буде випадковою, оскільки генерується детермінованим алгоритмом, але існує багато статистичних тестів, які можна

використовувати для того, щоб оцінити ступінь випадковості поведінки послідовності.

3. Функція генерації повинна ефективно реалізовуватись в рамках 32-бітної арифметики.

Усі ці три критерії можуть задовольнятись при адекватному виборі значень a, c та m. Відносно першого критерію можна довести, що якщо m є простим і c = 0, то для певних значень a період генерованої функцією послідовності виявляється рівним m-1 і в цій послідовності буде відсутнім тільки значення 0. В 32-бітній арифметиці зручним простим значенням для m є значення 2^{31} -1. В цьому випадку функція генерації приймає вигляд

$$X_{n+1} = (aX_n) \bmod (2^{31}-1)$$

З більш ніж двох мільйонів можливих значень a тільки декілька множників відповідають функції, що витримує усі три тести. Одним з таких значень є a = 7^5 = 16807, яке було знайдено і використано для родини комп'ютерів IBM 360. Відповідний генератор знайшов широке застосування, і тому він був підданий більш ретельному аналізу, ніж будь-який інший генератор псевдовипадкових чисел. Він нерідко рекомендується для статистичного та імітаційного моделювання різноманітних процесів. Перевагою алгоритму лінійного порівняння є те, що якщо вибрати адекватні множник та модуль порівняння, то створювана послідовність чисел виявляється статистично невідрізнимою від послідовності чисел, що вибираються випадково (але незворотно) з множини чисел 1, 2, …, m-1. Однак в самому алгоритмі немає нічого випадкового взагалі, крім вибору початкового значення X_0. Якщо це значення вибране, інші числа послідовності визначаються ним однозначно. Це виявляється дуже важливим з точки зору криптоаналізу. Якщо противник знає, що використовується алгоритм лінійного порівняння і якщо до того ж йому відомі параметри алгоритму (наприклад, a = 7^5 , c = 0, m = 2^{31} -1), то, відкривши усього одно число, противник може отримати всі наступні. Але якщо навіть опонент знає тільки те, що вибрано алгоритм лінійного порівняння, знання невеликої частини послідовності вже достатньо для того, щоб визначити усі параметри алгоритму. Припустимо, наприклад, що противник зможе визначити значення для X_0, X_1, X_2 та X_3. Тоді

$$X_1 = (aX_0 + c) \bmod m,$$
$$X_2 = (aX_1 + c) \bmod m,$$
$$X_3 = (aX_2 + c) \bmod m.$$

Ці рівняння можуть бути розв'язані відносно a, c та m.

Отже, хоча і зручно використовувати хороший генератор псевдовипадкових чисел, бажано подбати про те, що генерована послідовність була дійсно невідтворюваною, щоб знання частини послідовності не давало опоненту можливості визначити наступні елементи послідовності. Ця мета може бути досягнута цілим рядом способів. Наприклад можна змінювати потік псевдовипадкових чисел, використовуючи для цього системний час. Один зі способів на основі системного годинника полягає в ініціалізації нової послідовності після отримання кожних N чисел, використовуючи для початкового числа поточне значення часу (mod m). А можна просто додавати до кожного псевдовипадкового числа поточне значення часу (mod m).

Завдання до виконання роботи

Згідно до варіанту, наведеного в таблиці, створити програмну реалізацію генератора псевдовипадкових чисел за алгоритмом лінійного порівняння. Програма повинна генерувати послідовність із заданої при вводі кількості псевдовипадкових чисел, результати повинні як виводитись на екран, так і зберігатись у файл. Перевірити період функції генерації, зробити висновок про адекватність вибору параметрів алгоритму. У звіті навести протокол роботи програми, значення періоду функції генерації та зробити висновок про придатність цього генератора для задач криптографії.

2.	2^{11}-1	3^5	1	4

Хід роботи

1. Програмна реалізація:

```
c = 1
m = 2**11 - 1
a = 3**5
x = 4

primes = []


def gen_primes(x):
    if x <= len(primes):
        return primes

    i = max(len(primes), 2)
    while i <= x:
        prime = True
        for prime in primes:
            if i % prime == 0:
                prime = False
                break
        if prime:
            primes.append(i)
        i += 1

    return primes


def gen_primes_up_to(a):
    primes = gen_primes(a)
    retVal = []
    for prime in primes:
        if prime > a:
            break
        retVal.append(prime)
    return retVal


def gen_rand(x):
    return (x * a + c) % m


def factorize(a):
```

```
    factors = set()

    smallprimes = []

    limit = int(a ** 0.5) + 1
    pos_primes = gen_primes_up_to(limit)

    for prime in pos_primes:
        if a % prime == 0:
            smallprimes.append(prime)

    factors.update(smallprimes)
    for prime in smallprimes:
        factors.update(factorize(a / prime))
    return factors


def are_coprime(a, b):
    factors_a = factorize(a)
    factors_b = factorize(b)

    return factors_a.isdisjoint(factors_b)


def output(f, x):
    print(x)
    f.write(str(x) + "\n")


print("Analysing coefficients:")
c_m_coprime = are_coprime(c, m)
b = a - 1
m_factors = factorize(m)
b_is_multiple_of_m_factors = True
for factor in m_factors:
    if b % factor != 0:
        b_is_multiple_of_m_factors = False
        break
b_4_m_4 = True
if m % 4 == 0 and b % 4 != 0:
    b_4_m_4 = False

if c_m_coprime and b_is_multiple_of_m_factors and b_4_m_4:
    print("The period is full.")
else:
    print("The period can not be full.")


period = 0
with open("file.txt", 'w') as f:
    x1 = gen_rand(x)

    n = int(input("Please enter a number of random numbers to generate."))

    i = 0
    # while period != 0 and i < n:
    for i in range(n):
        i += 1
        x = gen_rand(x)
        if i < n:
            output(f, x)

        if i != 1 and x == x1 and period == 0:
```

```
        period = i

if period != 0 and n < m:
    print(
        "Period is less than m(and equals to",
        period,
        "), thus it is not full.")
```

2. Результат виконання програми

```
Analysing coefficients:
The period can not be full.
Please enter a number of random numbers to generate.10
973
1035
1772
727
620
1230
29
907
1373
```

file.txt - Notepad

File Edit Format View Help

```
973
1035
1772
727
620
1230
29
907
1373
```

Висновок: Під час виконання даної лабораторної роботи я ознайомився з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел та навчився створювати програмні генератори псевдовипадкових чисел для використання в системах захисту інформації.

Вибір значень для a, c та m є дуже важливим з точки зору створення хорошого генератора псевдовипадкових чисел. У випадку з моїм варіантом ми отримаємо послідовність наведену вище, яка не є задовільною, адже період не є цілим. Для перевірки цілосності періоду я використовував достатні умови: c і m – взаємопрості, a – 1 не кратний будь-якого з простих чисел факторізації m.

Для задач криптографії цей алгоритм не є придатним через те, що, якщо противник знає, що використовується алгоритм лінійного порівняння і якщо до того ж йому відомі параметри алгоритму, то, відкривши усього одне число, противник може отримати всі наступні.