УДК 511.2

КРИПТОГРАФІЧНІ ПЕРЕТВОРЕННЯ ЗА ДОПОМОГОЮ ПРОСТИХ ЧИСЕЛ

Дейно Євгенія Володимирівна
Науковий керівник доц. Петренко О.Є.
Харківський інститут банківської справи

Розвиток інформаційних технологій сприяв їх застосуванню в усіх сферах життя суспільства. Це привело до поширеного використання інформаційних технологій в електронному документообігу, електронній комерції, електронному банкінгу. В зв'язку з цим виникає задача забезпечити цілісність та конфіденційність інформації, яка передається інформаційно-телекомунікаційними системами. Для цього застосовують асиметричні та симетричні криптографічні перетворення. Цілісність та конфіденційність інформації в криптографічних примітивах ґрунтується на застосування одного або двох ключів. Процедура генерації, зберігання, знищення ключа вимагає проведенню серйозної роботи, тому що ключова інформація забезпечує можливість ефективної дії криптографічних примітивів для захисту конференційної інформації. Прості та псевдопрості числа відіграють важливу роль при генерації ключової інформації.

Актуальність роботи полягає в тому, що досі не знайдено справедливої математичної формули, яка б точно (при всіх допустимих значеннях аргументу) дозволяла знайти як завгодно велике просте число.

Мета роботи: знаходження найоптимальнішого способу генерації великих простих чисел, для подальшого застосування їх в криптографічних перетвореннях.

Прості способи знаходження початкового списку простих чисел аж до деякого значення дають решето Ератосфена, решето Сундарама і решето Аткіна. Однак, на практиці замість отримання списку простих чисел часто потрібно перевірити, чи є дане число простим. Алгоритми, що вирішують це завдання, називаються тестами простоти. Існує безліч поліноміальних тестів простоти, але більшість їх є імовірнісними (наприклад, тест Міллера - Рабіна) та використовуються для потреб криптографії.

Найпростіші фільтруючі пристрої відокремлюють прості числа шляхом ділення на 2, 3, 4 і т. д. Якщо ввести в пристрій число n, воно перевіряє його діленням на 2, на 3, на 4 і продовжує перевірки до тих пір, поки одна з них не виявиться успішною або дільник не досягне n. У першому випадку число складене, у другому - просте. Алгоритм цієї моделі може послужити основою найпростішої програми для домашнього комп'ютера. Назвемо цей алгоритм SLUICE 1:

input n
f= 1

```
for k= 2 to п - 1
test = rem (n/k)
if test = 0 then f - 0
if f = 1 then output «просте»
```

На вході програма приймає число *n*, що вводиться людиною з клавіатури. Потім програма встановлює для змінної (діє як прапорець) значення 1. Якщо f все ще дорівнює 1, коли програма завершує свої обчислення, отже, число є простим. У тілі циклу багаторазово повторюється один умовний оператор if. Індекс k пробігає значення від 2 до n-1. Для кожного значення k програма виконує ділення n/k, бере залишок від ділення (rem) і запам'ятовує його під ім'ям test. Найчастіше значення test виявляється не нульовим, іншими словами, k не ділить *n* без залишку. Але якщо хоч раз воно виявляється нульовим, програма негайно скидає прапорець f, записуючи туди 0, і це значення зберігається аж до кінця циклу. Якщо умова у другому операторі if задовольняється, програма друкує «просте». Якщо ж f було встановлено у нуль десь по ходу виконання циклу, то програма зберігає похмуре мовчання. Однак роботу програми SLUICE1 можна значно прискорити, якщо ввести в неї деякі зміни, але не має ніякого сенсу перевіряти, чи ділиться число *n* на k, якщо k більше квадратного кореня *n*, тому що принаймні один дільник числа *n*, якщо він є, не повинен перевершувати квадратного кореня *n*, а одного дільника вже достатньо, щоб вважати число складеним. Новий алгоритм, який ми назвемо SLUICE2, достатньою мірою відрізняється від свого нехитрого попередника і виглядає наступним чином:

```
r= 1
p(1)= 2
for n= 3 to 1000
k= 1
f=1
while f=1 and p(k )<= sqrt (n )
test=rem (n/p (k ))
if test =0 then f=0
k= k + 1
if f=1 then r= r + 1
P (r)= n
```

Оскільки програма SLUICE2 вимагає списку простих чисел, вона запам'ятовує кожне нове просте число в спеціальному масиві p. Змінна k є індексом, що вказує на останній елемент p. Таким чином, програма завжди знає, куди помістити наступне одержане нею просте число.

Висновки

У роботі були розглянуті різні види й методи пошуку простих чисел, знаходження найоптимальнішого способу генерації великих простих чисел, для подальшого застосування їх в криптографічних перетвореннях. Ця робота дозволила мені побачити світ простих цифр з іншого боку.

Незважаючи на те, що я зрозуміла недосконаліть сучасних знань, я рада тому, що можу внести свій вклад у вирішення цих проблем і завдань, і що на рівні сучасних технологій все ж є над чим замислитися.

Список використаних джерел.

1. Нестеренко Ю. В. «Алгоритмические проблемы теории чисел» // «Введение в криптографию» / Під редакцією В. В. Ященко. — Пітер, 2001.
2. Гальперін Г. «Просто о простых числах» // Квант. — 1987.- № 4. — С. 9-14,38.
3. Кордемський Б. А. «Математическая смекалка». — М.: ГИФМЛ, 1958.
4. Матіясевич Ю. «Формулы для простых чисел» // «Квант». — 1975. — № 5. — С. 5-13.