Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «Криптографические методы защиты информации»

Тема: «факторизация составного числа»

**Выполнил:**

Студент 2 курса

Группы ИИ-23

Романюк А.П.

**Проверил:**

Хацкевич А. С.

Брест 2024

**Цель:** Освоить простые алгоритмы факторизации составного числа

**Ход работы:**

**Вариант 9**



#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <vector>

#include "ap/ap.hpp"

ap\_uint<32768> long\_pow(ap\_uint<32768>a, ap\_uint<32768>b) {

ap\_uint<32768> result = 1;

for (int i = 0; i < b - 1; i++) {

result \*= a;

}

return result;

}

ap\_uint<32768> modulo(ap\_uint<32768>a, ap\_uint<32768>b) {

if (a < b) {

return a;

}

else {

return a - (a / b) \* b;

}

}

int NOD(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

std::vector<int> sieve\_of\_eratosthen(int n) {

std::vector<bool> is\_prime(n + 1, true);

std::vector<int> primes;

is\_prime[0] = is\_prime[1] = false; // 0 и 1 не являются простыми числами

for (int i = 2; i \* i <= n; ++i) {

if (is\_prime[i]) {

for (int j = i \* i; j <= n; j += i) {

is\_prime[j] = false;

}

}

}

for (int i = 2; i <= n; ++i) {

if (is\_prime[i]) {

primes.push\_back(i);

}

}

return primes;

}

bool rubber\_miller\_test(int n) {

if (n <= 3) return 1;

int t = n - 1;

int s = 0;

while (t % 2 == 0) {

t /= 2;

s += 1;

}

for (int k = 0; k < 5; k++) {

int a = rand() % (n - 3) + 2;

ap\_uint<32768> x = modulo(long\_pow(a, t), n);

if (x == 1 || x == n - 1) continue;

for (int b = 0; b < s - 1; b++) {

x = x \* x % n;

if (x == 1) return 0;

if (x == n - 1) break;

}

if (x != n - 1) return 0;

}

return 1;

}

int true\_rand\_num(int p = 8) {

while (true)

{

unsigned int random\_number = rand() % (1 << p) + 4;

random\_number |= 1u;

if (!rubber\_miller\_test(random\_number)) {

return random\_number;

}

}

}

std::vector<int> bruteforce\_factorization(int n, std::vector<int> primes) {

std::vector<int> result;

for (int num : primes) {

while (n % num == 0) {

n /= num;

result.push\_back(num);

}

}

return result;

}

int func(int x, int n) {

return ((x \* x) - x) % n;

}

std::vector<int> factorization(int n) {

std::vector<int> result;

int N = n;

while (n != 1) {

int x = rand() % (n - 1) + 1; // Изменение диапазона генерации случайного числа

int y = x;

int d = 1;

while (d == 1) {

x = func(x, n);

y = func(func(y, n), n);

d = NOD(abs(x - y), n);

}

if (d != N) {

n /= d;

if (!rubber\_miller\_test(d)) {

std::vector<int> factors\_of\_d = factorization(d); // Рекурсивная факторизация непростого делителя

result.insert(result.end(), factors\_of\_d.begin(), factors\_of\_d.end());

}

else {

result.push\_back(d);

}

}

}

return result;

}

int main() {

srand(time(NULL));

std::vector<int> primes = sieve\_of\_eratosthen(256);

int n = true\_rand\_num(16);

std::cout << n << "\n";

for (int num : bruteforce\_factorization(n,primes)) {

std::cout << num << " ";

}

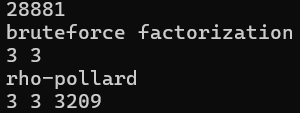
std::cout << "\n";

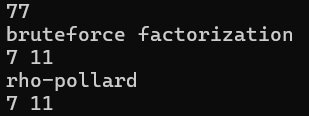
for (int num : factorization(n)) {

std::cout << num << " ";

}

}





**Вывод:** в ходе лабораторной работы я научился шифровать и сжимать информацию.