Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

Радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

по курсу “Средства и методы защиты информации в интеллектуальных системах”

Вариант 23

Выполнила:

Студент гр. 321702 Рублевская Е. А.

Проверил: Захаров В. В.

Минск

2025

**Тема:** Открытое распространение ключей

**Задание для самостоятельного выполнения:**

Для заданного простого P=1973 найти g – примитивный элемент конечного поля GF(P) и выполнить генерацию общего секрета. Для нахождения g воспользуйтесь методом перебора по возрастанию, возведения в степень по модулю P и проверки того факта, что все степени принимают значения от 0 до P – 1.

**Выполнение:**

**Задание 1**

**Код программы:**

#include <iostream>

#include <vector>

#include <cmath>

#include <set>

using namespace std;

// Функция для вычисления (a^b) % mod

long long powerMod(long long a, long long b, long long P) {

long long result = 1;

a = a % P;

while (b > 0) {

if (b % 2 == 1) {

result = (result \* a) % P;

}

b = b >> 1;

a = (a \* a) % P;

}

return result;

}

// Проверка, является ли число примитивным корнем

bool isPrimitiveRoot(int g, int p) {

set<int> powers;

for (int i = 1; i < p; i++) {

long long power = powerMod(g, i, p);

powers.insert(power);

}

return powers.size() == (p - 1);

}

// Поиск первого примитивного корня

int findPrimitiveRoot(int p) {

for (int g = 2; g < p; g++) {

if (isPrimitiveRoot(g, p)) {

return g;

}

}

return -1; // Если не найдено (но для простого числа всегда найдется)

}

// Генерация общего секрета

void generateSecret(int p, int g) {

// Выбираем случайные приватные ключи для Алисы и Боба

int a = 5; // приватный ключ Алисы (например)

int b = 7; // приватный ключ Боба (например)

// Алиса вычисляет A = g^a % p

long long A = powerMod(g, a, p);

// Боб вычисляет B = g^b % p

long long B = powerMod(g, b, p);

// Алиса вычисляет общий секрет: s\_A = B^a % p

long long s\_A = powerMod(B, a, p);

// Боб вычисляет общий секрет: s\_B = A^b % p

long long s\_B = powerMod(A, b, p);

cout << "Приватный ключ Алисы (a): " << a << endl;

cout << "Приватный ключ Боба (b): " << b << endl;

cout << "Открытый ключ Алисы (A): " << A << endl;

cout << "Открытый ключ Боба (B): " << B << endl;

cout << "Общий секрет (s\_A): " << s\_A << endl;

cout << "Общий секрет (s\_B): " << s\_B << endl;

}

int main() {

system("chcp 1251");

int P = 2971;

// Находим примитивный корень

int g = findPrimitiveRoot(P);

if (g == -1) {

cout << "Примитивный корень не найден." << endl;

return 1;

}

cout << "Примитивный элемент для поля GF(" << P << "): " << g << endl;

// Генерация общего секрета

generateSecret(P, g);

return 0;

}

**Основные функции программы:**

* powerMod – функция, которая вычисляет с помощью простого числа P и первообразного корня g специальное значение по формуле . Изначально результат равен 1. Мы берем a по модулю P, а потом, в цикле, проверяем, является ли b нечетным. Если это так, то результат умножается на текущее значение a и берется по модулю. После этого b делится пополам, а a возводится в квадрат и снова берется по модулю. Цикл продолжается, пока b не станет равным 0.
* isPrimitiveRoot – функция, которая проверяет, является ли число g примитивным корнем для поля GF(p). В цикле от 1 до p-1 для каждого I вычисляется . Результат сохраняется во множество, которое автоматически хранит только уникальные значения. И если количество уникальных значения равно p-1, то g является примитивным корнем
* findPrimitiveRoot – функция, которая находит первый примитивный корень g для поля GF(p). Она перебирает все значения g от 2 до p-1 и для каждого числа вызывается isPrimitiveRoot, чтобы проверить, является ли он примитивным корнем.
* generateSecret – функция, которая выводит в консоль все результаты.

**Результат программы:**

На рисунке 1 приведен первый примитивный корень для числа 1973, приватные ключи Алисы (5) и Боба (7), открытые ключи Алисы и Боба, которые были вычислены на основе приватных ключей, а также общий секрет, вычисленный на основе ключей Алисы и Боба, чтобы показать их совпадение:

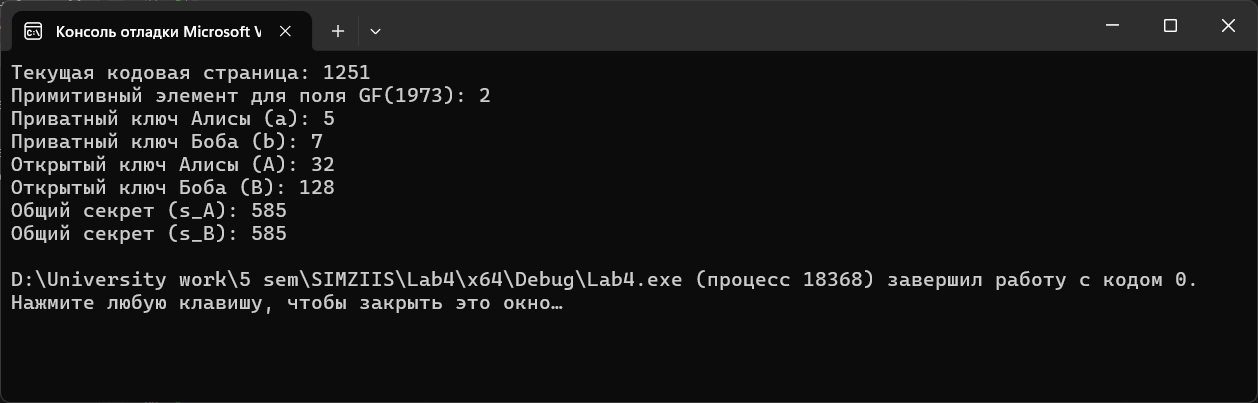
****

Рисунок 1. Отображение результатов программы.

**Задание 2**

Шаги, которые выполняют Алиса и Боб для вычисления общего секрета:

1. **Выбор параметров.** Оба участника соглашаются на использование общего простого числа P (в данном случае 2971) и примитивного корня g этого числа.
2. **Генерация приватных ключей.** Алиса и Боб выбирают случайные числа **a** и **b** в качестве своих приватных ключей.
3. **Вычисление открытых ключей.** Алиса вычисляет свой открытый ключ по формуле , а Боб вычисляет свой открытый ключ по формуле . Затем они обмениваются своими открытыми ключами.
4. **Вычисление общего секрета.** Получив открытый ключ В, Алиса вычисляет общий секрет . В свою очередь Боб тоже вычисляет общий секрет, получив открытый ключ Алисы по формуле . Оба вычисленных секрета будут равны, поскольку, .

**Вывод:**

В данной лабораторной работе было рассмотрено решение проблемы открытого распространения ключа. Для решения этой проблемы были придуманы односторонние функции, основной смысл которых заключался в том, f(x) для любого x из ее области определения легко вычислима, однако практически для всех y из области значений нахождение x вычислительно неосуществимо.

В результате были получены теоретические сведения о наиболее широко распространенных односторонних функциях, а именно:

- умножение и факторизация;

- возведение в квадрат и вычисление квадратного корня по модулю;

- возведение в степень по модулю и дискретное логарифмирование;

- криптографические хеш-функции.

В лабораторной работе была создана программа реализующая криптографический протокол Диффи-Хеллмана. Данная программа способна вычислить примитивный корень для определенного числа, а также на основе приватных ключей вычислить открытые ключи и общий секрет.

Если рассматривать данный протокол со стороны атакующего, то атакующий может перехватывать открытые ключи A и B, так как они передаются по незащищенному каналу. Однако сам по себе перехват открытых ключей не позволяет вычислить общий секрет без знания приватных ключей a и b. Для этого атакующий должен решить задачу дискретного логарифмирования, то есть найти a, что является крайне сложной задачей для больших чисел (проблема дискретного логарифма). Безопасность протокола зависит от размера простого числа Р. Чем больше Р, тем сложнее решить задачу дискретного логарифмирования. Для современных приложений рекомендуется использовать числа с длиной не менее 2048 бит, что делает вычисление дискретного логарифма практически невозможным для современного атакующего. В примере с Р=1973 длина числа составляет около 11 бит, что недостаточно для высокой безопасности и может быть взломано современными методами.

Возможные угрозы протоколу:

* **Угроза "Человек посередине"**. Атакующий может встать между Алисой и Бобом, перехватывая их сообщения и подставляя свои собственные открытые ключи. В этом случае атакующий устанавливает два отдельных секрета: один с Алисой, другой с Бобом, при этом оба участника думают, что делятся секретом друг с другом.
* **Атаки на малый модуль**. Если P слишком мало, то задача дискретного логарифмирования становится тривиальной для вычисления. Злоумышленник может попробовать все возможные значения и таким образом найти приватный ключ.
* **Атака на повторное использование ключей**. Если Алиса и Боб повторно используют свои приватные ключи a и b, это может облегчить атакующему задачу по нахождению общего секрета при многократных попытках атаки.

Возможная защита от угроз выше:

* **Угроза "Человек посередине"**. Можно использовать методы аутентификации. Алиса и Боб могут использовать цифровые подписи для удостоверения подлинности своих открытых ключей. Также можно использовать протоколы, основанные на сертификатах TLS, которые обеспечивают безопасный обмен ключами с использованием доверенных третьих сторон
* **Атаки на малый модуль**. Использование большого числа Р (не менее 2048 бит), а также выбор надёжных генераторов простых чисел и проверка на сильные криптографические свойства.
* **Атака на повторное использование ключей**. Каждый раз генерировать новые приватные ключи для сеанса. А также использовать протоколы с одноразовыми ключами.