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”

|  |  |
| --- | --- |
| Выполнил:  студент гр. 221702 | Юргилевич Е. В. |
| Проверил: | Крищенович В.А. |

Минск 2024

Лабораторная работа №4

ОТКРЫТОЕ РАСПРОСТРАНЕНИЕ КЛЮЧЕЙ

**Задание:** для заданного простого P (в соответствии с вариантом) 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.

**Вариант 20**

**P = 7883**

**Листинг программы для проверки g (первообразный корень по модулю P):**

int modularExponentiation(int base, int exp, int mod) {

int result = 1;

base = base % mod;

while (exp > 0) {

if (exp % 2 == 1) {

result = (result \* base) % mod;

}

exp /= 2;

base = (base \* base) % mod;

}

return result;

}

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

std::vector<int> factors;

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

if (n % i == 0) {

factors.push\_back(i);

while (n % i == 0) {

n /= i;

}

}

}

if (n > 1) {

factors.push\_back(n);

}

return factors;

}

bool isPrimitiveElement(int g, int p, const std::vector<int>& factors) {

for (int factor : factors) {

if (modularExponentiation(g, (p - 1) / factor, p) == 1) {

return false;

}

}

return true;

}

int main() {

int p = 7883;

int p\_minus\_1 = p - 1;

std::vector<int> factors = findPrimeFactors(p\_minus\_1);

std::cout << "Searching for a primitive element in F\_" << p << std::endl;

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

if (isPrimitiveElement(g, p, factors)) {

std::cout << "Primitive element found: " << g << std::endl;

break;

}

}

return 0;

}

**Описание основных функций программы:   
1) int modularExponentiation(int base, int exp, int mod)**

Эта функция вычисляет значение baseexp mod P помощью метода последовательного возведения в степень и умножения. Функция использует метод "возведения в квадрат и умножения" для эффективного вычисления степени. Она последовательно возводит основание в квадрат и умножает результат на текущую степень при необходимости. Модуль вычисляется на каждом шаге для предотвращения переполнения.

**2) std::vector<int> findPrimeFactors(int n)**

Данная функция находит все простые делители числа.

3) **bool isPrimitiveElement(int g, int p, const std::vector<int>& factors);**

Эта функция проверяет, является ли элемент g примитивным элементом конечного поля Fp . Для проверки примитивности функция вычисляет g(P-1)/d mod P для каждого делителя d из вектора простых делителей factors. Если для любого делителя результат равен 1, это означает, что g не является примитивным элементом. Если ни одно из вычислений не равно 1, g является примитивным элементом.

**Листинг программы для вычисления Ba mod P = gab mod P и Ab mod P = gab mod**

#include <iostream>

int modularExponentiation(int base, int exp, int mod) {

int result = 1;

base = base % mod;

while (exp > 0) {

if (exp % 2 == 1) {

result = (result \* base) % mod;

}

exp /= 2;

base = (base \* base) % mod;

}

return result;

}

int main() {

int g = 2;

int P = 7883; gt

int a, b;

std::cout << "Enter value for a: ";

std::cin >> a;

std::cout << "Enter value for b: ";

std::cin >> b;

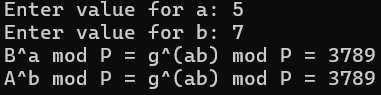
int gab\_mod\_P = modularExponentiation(g, a \* b, P);

std::cout << "B^a mod P = g^(ab) mod P = " << gab\_mod\_P << std::endl;

std::cout << "A^b mod P = g^(ab) mod P = " << gab\_mod\_P << std::endl;

return 0;

}

Умножение происходит через ту же функцию, что и в предыдущей программе. Результат выполнения для a = 5, b = 7:   
 

**Описание шагов, выполняемых участниками протокола – Алисой и**

**Бобом для вычисления общего секрета.**

1. Алиса и Боб выбирают g и P, где P — большое простое число (7883), g — первообразный корень по модулю P ( в нашем случае это число 2);
2. Затем Алиса и Боб выбирают секретный ключ a и b. Каждый знает только свой ключ и держит его в секрете. Пусть ключ Алисы будет равен 5, ключ Боба – 7;
3. Далее Алиса вычисляет ga mod P (25 mod 7883) и получает число 32 ;
4. Аналогично Боб вычисляет gb mod P (27 mod 7883) и получает  
   число 128 ;
5. Сейчас Алиса и Боб обмениваются числами, которые они получили;
6. Теперь Алиса вместо первообразного корня по модулю P в формулу ga mod P подставляет число, которое получила от Боба ( 1285 mod 7883), и получает число 3789 ;
7. Аналогично Боб вместо первообразного корня по модулю P в формулу gb mod P подставляет число, полученное от Алисы (327 mod 7883), и получает число 3789

;

1. Алиса и Боб получили секретный ключ: число 3789.

**Модель атакующего**: В этой модели предполагается, что атакующий обладает следующими возможностями:

* **Знание параметров**: Атакующему известно, что в системе используется конечное поле с модулем P=7883 и примитивный элемент g=2;
* **Доступ к передаваемым данным**: Атакующий может наблюдать за передачей значений Ba mod P и Ab mod P, но не знает секретных чисел a и b;
* **Алгоритм атакующего**: Атакующий может попытаться найти секретные значения a и b через методы перебора (brute-force), анализа или использования дискретного логарифма.

**Методы атакующего**:

* **Атака методом грубой силы (brute-force)**: Атакующий может попытаться перебрать возможные значения a или b и вычислить соответствующие значения ga mod P или gb mod P, чтобы в итоге восстановить ключ;
* **Анализ дискретного логарифма**: Если атакующий сможет решить задачу нахождения дискретного логарифма, то он сможет восстановить секретное число a или b из открытых данных Ba mod P или Ab mod P.

**Оценка безопасности ключа**:

* **Число возможных ключей**: Ключ K = gab mod P может принимать любые значения от 1 до P−1, то есть существует P−1=7882 возможных значений для ключа.
* **Безопасность**: Длина ключа в битах зависит от числа возможных ключей. Для числа P=7883 длина ключа будет приближенно равна: длина ключа (в битах) = log2(P) ≈ log2(7883) ≈ 12.96 бит. То есть длина ключа составляет примерно 13 бит. Это сравнительно короткий ключ, который может быть уязвим к современным методам атак, например, к методу дискретного логарифма или атаке грубой силы.

**Оценка сложности атаки**:

**Brute-force атака**: Если атакующий пытается перебрать все возможные ключи, ему потребуется проверить до P−1 возможных значений, что делает атаку с использованием грубой силы довольно простой для поля с модулем 7883. Для защиты от такой атаки рекомендуется подбирать большие числа a и b.