Министерство образования Республики Беларусь

Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Дисциплина: Средства и методы защиты информации в интеллектуальных системах

Отчёт к лабораторной работе №4

Выполнил: Потоцкий Д.А.

Группа: 221702

Проверила: 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 17**

**P = 8233**

### Описание основных функций программы: 1. is\_primitive(g, P)

Этот метод проверяет, является ли число g примитивным элементом в конечном поле GF(P). В контексте конечных полей, примитивный элемент — это такой элемент, который может генерировать все остальные ненулевые элементы поля при возведении в различные степени. Другими словами, это элемент, который порождает мультипликативную группу поля. Метод принимает два аргумента: число g, которое нужно проверить, и порядок поля P. Он возвращает True, если g является примитивным элементом, и False в противном случае. Этот метод важен для криптографии, потому что примитивные элементы играют ключевую роль в таких протоколах, как Диффи-Хеллман.

**2. find\_primitive(P)**

Этот метод предназначен для поиска примитивного элемента в конечном поле GF(P). Он принимает единственный аргумент P — порядок поля, и возвращает генератор, который перебирает возможные кандидаты для примитивных элементов. Он не просто находит первый примитивный элемент, а может возвращать несколько возможных вариантов, если они существуют. Это полезно в случае, если нам нужен не конкретный примитивный элемент, а любой из возможных. В криптографических протоколах, таких как Диффи-Хеллман, этот метод позволяет гибко выбирать примитивные элементы для генерации ключей.

**3. prime\_factors(n)**

Этот метод вычисляет и возвращает список простых делителей числа n. Простые делители — это такие числа, которые делят n без остатка и сами являются простыми числами. Метод полезен в тех случаях, когда требуется разложить число на множители, например, для проверки условий в теории чисел или криптографии. В данном случае этот метод используется для нахождения делителей порядка мультипликативной группы поля GF(P), что важно для проверки примитивности элемента.

**4. generate\_shared\_secret(P, a, b, random\_g=False)**

Этот метод реализует протокол Диффи-Хеллмана для генерации общего секрета между двумя сторонами (например, Алисой и Бобом). Протокол Диффи-Хеллмана позволяет двум сторонам, используя свои секретные ключи и общий открытый элемент (примитивный элемент g), вычислить общий секрет, который может использоваться для шифрования сообщений. Метод принимает порядок поля P, секретные ключи двух сторон a и b, а также флаг random\_g, который определяет, будет ли использоваться случайный примитивный элемент или первый найденный. Результатом метода является примитивный элемент и общий секрет, который вычисляется обеими сторонами.

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

* **Знание параметров**: Атакующему известно, что в системе используется конечное поле с модулем P=8233 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=8233 возможных значений для ключа.
* **Безопасность**: Длина ключа в битах зависит от числа возможных ключей. Для числа P=8233 длина ключа будет приближенно равна: длина ключа (в битах) = log2(P) ≈ log2(8233) ≈ 13 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 8233. Для защиты от такой атаки рекомендуется подбирать большие числа a и b.