**Учреждение образования**

**«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»**

**Кафедра интеллектуальных информационных технологий**

**Лабораторная работа №4 по курсу «СиМЗИИС» на тему:**

**«Открытое распространение ключей»**

**Вариант 7**

Выполнил студент: Исамиддинов Ботир

Группы: ***121731***

Проверил: Сальников Д.А

**МИНСК**

2023

**Задание**

**Вариант - 7**

Для заданного простого P (в соответствии с вариантом P = 8699) 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

Отчет должен содержать:

1) Листинги программ:

а) для проверки g (первообразный корень по модулю P);

б) для вычисления и .

2) Описание шагов, выполняемых участниками протокола – Алисой и Бобом для вычисления общего секрета.

4) Выводы, содержащие:

а) модель атакующего и оценки длины ключа;

б) возможные угрозы протоколу и предложения по защите от них.

Важно!

Возведение в степень выполнять методом последовательного возведения в квадрат и умножения.

При возведении в степень по модулю операцию взятия по модулю выполнять на каждом шаге возведения в квадрат и умножения для исключения переполнения.

**Задание 1**

Код программы будет предоставлен ниже.

Поле *GF*(*P*), где *P*=8699, требует нахождения примитивного элемента *g*, такого что все степени *gi* (где *i* принимает значения от 0 до *P*−1) различны.

Начнем с выбора *g*=2 и будем проверять его степени по модулю *P*=8699:

…..

1. Если все значения от 1 до P−1 включительно встречаются, то g - примитивный элемент. В данном примере, g=2 не является примитивным элементом.

Повторим этот процесс для g=3, g=4, и так далее, пока не найдем примитивный элемент.

1. Вычисление общего секрета:

После нахождения примитивного элемента g, каждый из участников (Алиса и Боб) выбирает свой собственный секретный ключ (*a* для Алисы, *b* для Боба). Эти секретные ключи затем используются для вычисления публичных ключей и общего секрета:

Публичные ключи:

Общий секрет:

Стоит заметить, что и у Алисы, и у Боба, один и тот же секретный ключ *a* и *b* используется для вычисления общего секрета.

Результат работы программы поиска примитивного элемента и генерации секретов приведен на рисунке 1.

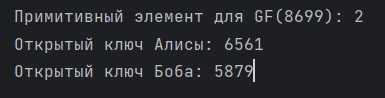


Рис. 1.

**Выводы:**

а) Модель атакующего и оценки длины ключа:

По условиям задачи, P = 8699. Процесс поиска примитивного элемента требует перебора значений g от 2 до P-1. Это может быть вычислительно затратной задачей, особенно для больших значений P. Длина ключа в данном случае равна битовой длине простого числа P.

б) Возможные угрозы протоколу и предложения по защите:

* **Атака перебором (brute force):** Если злоумышленник перехватит общедоступные ключи Алисы и Боба, он может попытаться перебрать все возможные значения секретного ключа. Для защиты от этого рода атаки, можно использовать большие значения P, что усложнит перебор.
* **Атака посредника (man-in-the-middle):** Если злоумышленник может перехватить и модифицировать общедоступные ключи, он сможет расшифровать и перехватывать сообщения. Для защиты от этого типа атак, необходимо использовать аутентификацию или подпись сообщений.
* **Атака воспроизведения (replay attack):** Злоумышленник может повторно отправить ранее перехваченные сообщения, что приведет к повторному использованию секретного ключа. Для защиты от этого типа атак, необходимо включать метки времени или случайные значения в обмен данных.
* **Атака компрометации приватного ключа:** Если приватный ключ Алисы или Боба будет скомпрометирован, все сообщения будут уязвимы. Поэтому, ключи должны храниться в безопасном месте и использоваться только авторизованными пользователями.

В целом, для обеспечения безопасности протокола Диффи-Хеллмана, следует использовать большие простые числа, хранить приватные ключи в надежных местах, и принимать меры против различных типов атак.

**Код программы.**



def is\_primitive\_element(g, p):  
 elements = set()  
 result = 1  
 for \_ in range(p - 1):  
 result = (result \* g) % p  
 elements.add(result)  
 return len(elements) == p - 1  
  
def find\_primitive\_element(p):  
 for g in range(2, p):  
 if is\_primitive\_element(g, p):  
 return g  
  
p = 8699  
g = find\_primitive\_element(p)  
print(f"Примитивный элемент для GF({p}): {g}")  
  
def calculate\_shared\_secret(private\_key, g, p):  
 return (g \*\* private\_key) % p  
  
# Пример приватных ключей для Алисы и Боба  
private\_key\_alice = 546  
private\_key\_bob = 874  
  
shared\_secret\_alice = calculate\_shared\_secret(private\_key\_alice, g, p)  
shared\_secret\_bob = calculate\_shared\_secret(private\_key\_bob, g, p)  
  
print(f"Открытый ключ Алисы: {shared\_secret\_alice}")  
print(f"Открытый ключ Боба: {shared\_secret\_bob}")