Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Кафедра прикладной математики и информатики

Отчет по лабораторной работе №1 на тему
"RSA"
по дисциплине
"Дискретная математика"

Выполнил студент гр. 5030102/20201 Фатахов Т.М.

Санкт-Петербург
2024

1 Задание

Цель лабораторной работы — реализовать алгоритм шифрования с открытым ключом RSA, который включает следующие этапы:

- Генерацию ключей — открытого (пара чисел (n, e)) и закрытого (число d);
- Шифрование сообщения с использованием открытого ключа;
- Расшифрование сообщения с использованием закрытого ключа.

Реализация должна использовать случайные простые числа длиной не менее 1024 бит. Ключи сохраняются в отдельных файлах, поддерживается работа с большими числами. В дополнение к требованиям реализованы методы генерации простых чисел, возведения в степень по модулю, теста Миллера-Рабина для проверки простоты, а также алгоритмы Евклида и расширенного Евклида для нахождения обратного по модулю.

2 Язык программирования

Реализация выполнена на языке программирования Python (версия 3.10). Для работы с большими числами использовалась встроенная поддержка Python, что исключает необходимость в сторонних библиотеках для этой задачи.

3 Описание алгоритмов

3.1 Генерация случайных простых чисел

Для генерации простых чисел используется следующий подход:

1. Генерируется случайное число длиной 1024 бита.
2. Выполняется проверка простоты с использованием теста Миллера-Рабина.
3. В случае если число не является простым, процедура повторяется до нахождения простого числа.

3.2 Генерация ключей RSA

Генерация ключей состоит из следующих шагов:

1. Выбираются два случайных простых числа p и q.
2. Вычисляется $n = p \times q$, первая часть открытого ключа.
3. Вычисляется $\varphi(n) = (p - 1)(q - 1)$.
4. Определяется открытая экспонента e, которая является стандартной константой, равной 65537.

5. Вычисляется закрытая экспонента d как обратное к e по модулю $\varphi(n)$ с использованием расширенного алгоритма Евклида.

3.3 Шифрование

Для шифрования сообщения M (в данном случае, строки, преобразованной в большое число):

$$C = M^e \mod n$$

Результат C сохраняется в файл.

3.4 Расшифрование

Процесс расшифрования использует частное значение d:

$$M = C^d \mod n$$

Затем результат переводится обратно в строку.

3.5 Алгоритмы, используемые в RSA

- **Тест Миллера-Рабина**: Используется для вероятностной проверки простоты чисел.
- **Быстрое возведение в степень по модулю**: Позволяет выполнять возведение в степень для больших чисел в рамках заданного модуля.
- **Алгоритм Евклида и расширенный алгоритм Евклида**: Позволяет находить обратное по модулю и определяет d, соответствующее e.

4 Демонстрация работы на примере

Рассмотрим генерацию ключей и шифрование небольшого сообщения на тестовых данных. Пусть $p = 3$, $q = 11$, тогда:

- $n = 3 \times 11 = 33$;
- $\varphi(n) = (3 - 1)(11 - 1) = 20$;
- $e = 7$, выбранное как взаимно простое с $\varphi(n)$.
- $d = 7^{-1} \mod 20 = 3$.

Пусть сообщение $M = 2$. Тогда:

- Шифротекст $C = M^e \mod n = 2^7 \mod 33 = 29$.
- Расшифрованное сообщение $M = C^d \mod n = 29^3 \mod 33 = 2$, что совпадает с исходным сообщением M.

5 Область применения и возможные ошибки

RSA шифрование с открытым ключом широко применяется для обеспечения конфиденциальности передачи данных. Алгоритм надёжен, если выполняются требования к длине ключей (от 2048 бит) и соблюдаются принципы безопасности ключей. Ошибки могут возникнуть в следующих случаях:

- Несоблюдение длины ключа приводит к уязвимости RSA к атакам;
- Неправильная работа с файлами ввода/вывода может привести к некорректной загрузке ключей и данных;
- Использование малых простых чисел, как в данном примере, уязвимо для подбора значений ключа.

6 Формат входных и выходных данных

- **Входные данные**: Файлы с сообщением и ключами.
- **Выходные данные**: Файлы, содержащие зашифрованное и расшифрованное сообщение.
- Открытый ключ хранится в текстовом файле в виде чисел e и n в отдельной строке.
- Закрытый ключ хранится в текстовом файле в виде чисел d и n в отдельной строке.

7 Дополнительные требования

Дополнительные требования, указанные в задании, включают:

- Программа полностью реализована на Python;
- Программа генерирует случайные 1024-битные простые числа, с которыми работают функции RSA.