Санкт-Петербургский Политехнический Университет Петра Великого

Физико-механический институт

Кафедра Прикладной математики и Информатики

**Отчет по Лабораторной работе №1**

Тема: Кодировка методом RSA

Предмет: Дискретная математика

Студент: Егоркин Станислав Дмитриевич

Группа: 5030102/20202

Преподаватель: Нахатович Михаил Алексеевич

1. **Требования:**

Реализовать алгоритм кодировки RSA. Кроме операций шифрования по открытому ключу и расшифрования по закрытому ключу необходимо поддержать операцию генерации ключей и сохранения их в отдельном файле или файлах. Случайные простые числа, генерируемые при генерации ключей, должны содержать не менее 1024 бит. Генерация случайных простых чисел, выбор числа *e* (может быть и константа), поиск числа *d*, возведение в степень с последующим делением по модулю. Необходимо шифровать сообщение полностью, а не по символам.

* Программа реализована на языке C++ с использованием библиотеки boost для работы с большими числами

1. **Идея алгоритма:**

Использования произведения больших простых чисел для создания общего ключа и кодировки сообщения. Безопасность такого метода кодировки заключается в почти невозможном (на данный момент) получении двух простых больших числа из их произведения для дешифрации информации.

1. **Алгоритмы программы:**
2. Создание больших чисел -

Генерируется случайное число длинной 1024 бит, минимальное значение которого равно 21024 . Далее проверяется его простота. Если условие не выполнено, то генерация продолжается до тех пор, пока не выполнится условие по тесту Миллера-Рабина.

1. Тест Миллера-Рабина

Этот вероятностный тест на простоту. Алгоритм:

1. Дано n, нужно найти s, такое что n – 1 = 2Sq для некоторого нечетного q.

2. Возьмем случайное a ∈ {1,...,n−1}

3. Если aq = 1, то n проходит тест и мы прекращаем выполнение.

4. Для i= 0, ... , s−1 проверить равенство a(2^i)q = −1. Если равенство выполняется, то n проходит тест (прекращаем выполнение).

5. Если ни одно из вышеприведенных условий не выполнено, то n – составное.

1. Нахождение обратного по модулю работает, основываясь на расширенном алгоритме Евклида (Реализации алгоритма не приводится так как считается общеизвестной)
2. Алгоритм быстрого возведения в степень:

Вход [a,n,b]

Res = a

если n – четно, то res = (res\*res) mod(b)

иначе res = (res\*a) mod(b)

n//2 – (Целочисленное деление)

Выход[res]

Таким образом за log(n) мы получаем ответ

1. **Описание работы RSA:**
2. Генерация ключей (generate\_keys):

В данной функции генерируются приватный и открытый ключ (p, q – через создание больших чисел, n = p \* q, phi(n) = (p-1) \* (q-1), е = 65537 (константа), взаимно простая с phi, d – обратное к e по модулю phi)

1. Кодировка (encrypt):

Каждый символ сообщения переводиться в число и, умножаясь на 256 (максимальный код элемента в кодировке ASCII), добавляется к переменной.

1. Декодирование (decrypt):

Из переменной, в которой храниться закодированное сообщение в виде числа, берётся остаток от деления этого числа на 256 и переводиться в символ.

1. **Пример работы алгоритма:**

Генерируем два случайных больших числа p и q. Для примера работы алгоритма возьмем 661 и 677 соответственно и проверим их на простоту.

Проверяем их на простоту с помощью теста Миллера-Рабина:

1. Проверяем справедливость хотя бы 1 из условий ( ad = 1 mod n или  a(2^i)q = −1. Если выполняется, то число простое, иначе составное:
2. Разложим число n-1 в виде 2s\*d:

Для p=661:

p−1=660.

Разложение 660 =22⋅165(s=2, d=165).

Для q=677:

q−1=676.

Разложение 676=22⋅169 (s=2, d=169).

1. Выберем случайное основание:

a ∈ [1,...,n−1]

Для p= 661 a=7;

Для q=667 a=13.

1. Вычисление x=ad mod  n:

Для p = 661, a=7, d=165:

x = 7165 mod 661 = 555

Для q=677, a=13, d=169:

x=13169 mod 677 = 1. – условие выполнено – 677 простое

1. Возводим x в квадрат i=[1, s-1] раз

x = 7165\*2 mod 661 =-1 – условие выполнено – 661 простое.

N=p\*q=447497.

Вычислим d через алгоритм нахождение обратного по модулю:

d\*e=1( mod phi )

phi=(p-1)(q-1) = 446160

Делим phi на e, потом целое число на остаток, пока остаток не будет равен 1:

446160=6⋅65537+49782.

65537=1⋅49782+15755.

49782=3\*15755+2532

15755=6\*2532+467

2532=5\*467+167

467=2\*167+133

167=1\*133+34

133=3\*34+31

34=1\*31+3

31=10\*3+1

После этого в обратную сторону находим d:

1=31-10\*3=31-10(34-1\*31)=11\*31-10\*34=11\*(133-3\*34)-10\*34=11\*133-43\*34=…= 127298⋅65537−21329⋅446160, где

127298 – это значение d, но больше, чем phi, поэтому берем остаток :  
d= 127298 mod 446160 = 40193.

Получили ключи {40193,447497}, {65537, 447497}

Вход: «A2»

1. Преобразование строки в число

Ans = 65 + 50\*256 = 12865 ( «A» = 65 ASCII)

1. Шифрование

Вызываем алгоритм возведения в степень по модулю для Ans и открытого ключа.

Ans^e(mod N) = 12865^40193 (mod , 447497)= 51431

Это и есть зашифрованное сообщение – 51431

1. Расшифровка

Вызываем алгоритм возведения в степень по модулю для шифра и приватного ключа. Далее преобразование числа в строку:

51431^65537 (447497) = 12865

12865 %256 = 65

Записываем в строчку «А»

25185/256=50

50%256 = 50

Добавляем в строчку «2»

Вывод – «А2»

1. **Формат данных:**

Код может реализовывать как генерацию ключей, шифровку и расшифровку одновременно, так и каждый процесс отдельно. Если необходима генерация ключей, то необходимо ввести только текст. В данном случаи код выводит только публичный и приватный ключи. При шифровке текста необходим текст и публичный ключ. В данном случаи код выводит зашифрованный текст. При дешифровке текста необходим зашифрованный текст и приватный ключ. В данном случаи выводится расшифрованный текст. При необходимости всех процессов необходимо только текст для зашифровки.

Текст для шифровки должен быть записан в формате ASCII.

Зашифрованный текст должен быть записан в формате натурального числа.

Публичный ключ должен быть записан в формате двух чисел (константа e и произведения двух больших натуральных чисел n)

Приватный ключ должен быть записан в формате двух чисел (секретной экспоненты d и произведения двух больших натуральных чисел n)

1. **Ошибка метода:** При использовании теста Милера-Рабина вероятность ошибки, когда доказательство показывает ошибочно простое число, вместо составного, составляет минимум ¼, но при использовании теста k раз вероятность уменьшается до 2-2k. В своей программе тест Милера-Рабина проходится 25 раз, чтобы уменьшить вероятность ошибки для каждого числа до 2-50 .
2. **Применение констант:** Значение e=65537 широко используется благодаря своей эффективности. Оно минимизирует количество операций при шифровании, сохраняя при этом устойчивость к атакам. Длина в 1024 бита обеспечивает высокий уровень стойкости. Также эти константы соответствуют общепринятым стандартам, включая рекомендации NIST для алгоритмов асимметричного шифрования.
3. **Вывод:**

Данный алгоритм затратен по времени при работе с большими числа и имеет недостаток, заключающийся в использовании теста Миллера-Рабина для проверки простоты чисел.

Источники информации:

1. <https://habr.com/ru/articles/485872/>
2. <https://en.wikipedia.org/wiki/RSA_(cryptosystem)>