МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное

учреждение высшего образования

ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук

имени И. И. Воровича

Направление подготовки

02.03.02 – Фундаментальная информатика

и информационные технологии

«Алгоритм RSA и его реализация с применением средств библиотеки NTL»

Выпускная квалификационная работа

на степень бакалавра

Студента 4 курса

А. В. Копнина

Научный руководитель:

д.ф.-м.н., проф., зав. каф. В. С. Пилиди

Допущено к защите:

Руководитель направления ФИИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Пилиди

Ростов-на-Дону

2017

**Содержание**

Введение ........................................................................................................... 3

Описание алгоритма ....................................................................................... 3

Создание двух ключей: открытого и закрытого ................................. 3

Шифрование ........................................................................................... 4

Декодирование ....................................................................................... 5

Криптостойкость алгоритма ........................................................................... 5

Реализация алгоритма RSA с применением средств библиотеки NTL для работы с большими целыми числами на языке C++ .................................... 6

Результаты работы программы .................................................................... 10

Список использованной литературы ........................................................... 12

**Введение**

Алгоритм RSA (название образовано от фамилий создателей алгоритма – Rivest, Shamir, Adleman) – криптографический алгоритм с открытым ключом, который основывается на вычислительной сложности задачи факторизации больших целых чисел.

RSA был разработан тремя учёными Массачусетского Технологического института (MIT) Рональдом Ривестом, Ади Шамиром и Леонардом Адлеманом в 1978 году. В настоящее время RSA является одной из наиболее распространённых ассиметричных криптосистем. RSA – первый полноценный алгоритм с открытым ключом, который может работать как в режиме шифрования данных, так и в режиме электронной цифровой подписи.

**Описание алгоритма**

Алгоритм RSA состоит из нескольких этапов:

**1) Создание двух ключей: открытого и секретного**

1. Для начала необходимо выбрать два различных случайных простых числа p и q.

2. Вычисление их произведения: . Число n будем называть модулем.

3. Затем требуется вычислить значение функции Эйлера от числа n.

Функция Эйлера – арифметическая функция, равная количеству натуральных чисел, меньших n и взаимно простых с ним. , где p, q – простые числа. Функция Эйлера от простого числа вычисляется по формуле . Но поскольку функция Эйлера мультипликативна, то .

4. Необходимо выбрать число e такое, что , .

Обычно выбирают простое число, которое содержит небольшое количество единичных бит в своём двоичном представлении. Например, простые числа Ферма. При этом стоит помнить, что слишком малые значения числа e могут ослабить безопасность алгоритма.

Число e называется открытой экспонентой.

5. Вычисляется число d – секретная экспонента. Число d мультипликативно обратное к числу e по модулю (n), то есть .

Обычно секретную экспоненту вычисляют при помощи расширенного алгоритма Евклида.

6. Ключи вычислены. Пара публикуется в качестве открытого ключа алгоритма RSA. А пара – закрытый ключ. Его держат в секрете.

**2) Шифрование**

После того, как мы получили открытый и закрытый ключи, можно переходить к процедуре шифрования сообщения.

Берём открытый ключ и текст сообщения m. Производим шифрование по формуле и получаем код, который можно передавать.

**3) Декодирование**

У нас имеется зашифрованное сообщение c, а также закрытый ключ . Исходное сообщение можно получить по формуле .

**Криптостойкость алгоритма**

Сложность нахождения секретного ключа зависит от сложности разложения числа n на простые множители. Числа p и q надо выбирать так, чтобы задача разложения n была достаточно сложна в вычислительном плане.

Для чисел p и q существует ряд рекомендаций.

1) Числа p и q должны быть достаточно большими. При этом они не должны сильно отличаться друг от друга, но и не должны быть слишком близкими друг к другу.

2) Числа p и q должны быть такими, чтобы наибольший общий делитель чисел и был небольшим. Рекомендуется, чтобы был равен 2.

3) p и q должны быть сильно простыми числами.

Сильно простым числом называется достаточно большое простое число s такое, что имеет большой простой делитель, а число имеет большой простой делитель t такой, что также обладает достаточно большим простым делителем.

Если хотя бы одна из этих рекомендаций не выполняется, то существует большая вероятность того, что n можно будет разложить на множители с помощью эффективных алгоритмов разложения.

В современной практике для обеспечения криптостойкости алгоритма RSA обычно используются числа p и q порядка 21024 и n порядка 22048.

**Реализация алгоритма RSA с применением средств библиотеки NTL для работы с большими целыми числами на языке C++**

Диапазоны стандартных типов целых чисел языка C++ (int, long, long long) крайне малы для безопасного использования алгоритма RSA, поскольку они не позволяют сгенерировать достаточно большие простые числа, чтобы обезопасить передачу данных. Как отмечено выше, необходимо работать с целыми числами размера 2048 бит. А со стремительным развитием производительности машин, в скором времени и такие большие числа не смогут обеспечить безопасность. Поэтому в реализации алгоритма RSA была использована библиотека NTL.

NTL – высокопроизводительная портируемая библиотека, написанная на языке C++, реализующая некоторые структуры данных (векторы, матрицы, полиномы) над целыми числами и над полями Галуа, а также позволяющая работать с целыми числами произвольной длины.

Для работы с такими числами из библиотеки NTL (тип ZZ) подключаем заголовочный файл NTL//ZZ.h. Также нам потребуется заголовочный файл *time.h* для генерации псевдослучайных чисел в зависимости от текущего времени на устройстве.

Функция GenPrimes будет генерировать два простых числа p и q заданного размера sz средствами библиотеки NTL (функция GenPrime). Затем будет вычисляться их произведение и значение функции Эйлера.

*void GenPrimes(ZZ& p, ZZ& q, ZZ& n, ZZ& eulerN, int sz)*

*{*

*GenPrime(p, sz);*

*GenPrime(q, sz);*

*n = p \* q;*

*eulerN = (p - 1) \* (q - 1);*

*}*

Функция GenKeys предназначается для генерации открытой экспоненты e (функция GenPrime из NTL) и вычисления по ней секретной экспоненты d (функция InvMod).

*void GenKeys(ZZ& e, ZZ& d, ZZ& eulerN, long sz)*

*{*

*do*

*GenPrime(e, sz);*

*while (e >= eulerN && GCD(e, eulerN) != 1);*

*InvMod(d, e, eulerN);*

*}*

Вычисленные открытый и закрытый ключи будут сохраняться в текстовый файл формата txt с помощью функции SaveKey.

*void SaveKey(ZZ& e, ZZ& d, ZZ& n)*

*{*

*fstream fpublic, fprivate;*

*fpublic.open("PublicKey.txt", ios::out);*

*fprivate.open("PrivateKey.txt", ios::out);*

*if (!fpublic || !fprivate)*

*{*

*cout << "Ошибка записи в файл. " << endl;*

*system("pause");*

*exit(0);*

*}*

*fpublic << e << "\n\n" << n;*

*fprivate << d << "\n\n" << n;*

*fpublic.close();*

*fprivate.close();*

*}*

Для того, чтобы можно было зашифровать не только число, но и текстовое сообщение, будем использовать функции StringToZZ и ZZToString.

*ZZ StringToZZ(string str)*

*{*

*ZZ number = conv<ZZ>(str[0]);*

*long len = str.length();*

*for (long i = 1; i < len; i++)*

*{*

*number \*= 128;*

*number += conv<ZZ>(str[i]);*

*}*

*return number;*

*}*

*string ZZToString(ZZ num)*

*{*

*long len = ceil(log(num) / log(128));*

*char\* str = new char[len];*

*for (long i = len - 1; i >= 0; i--)*

*{*

*str[i] = conv<int>(num % 128);*

*num /= 128;*

*}*

*str[len] = '\0';*

*return (string)str;*

*}*

Функции шифрования (Encryption) и дешифрования (Decryption), реализованные с помощью функции PowerMod (возведение в степень по модулю).

*ZZ Encryption(string str, ZZ& e, ZZ& n)*

*{*

*ZZ zz\_from = StringToZZ(str);*

*return PowerMod(zz\_from, e, n);*

*}*

*string Decryption(ZZ& cipherText, ZZ& d, ZZ& n)*

*{*

*return ZZToString(PowerMod(cipherText, d, n));*

*}*

Во всех функциях, реализующих алгоритм RSA, параметры передаются по ссылке, чтобы не передавать огромные числа в функцию по значению, замедляя время работы программы.

Константами SZ\_PRIME и SZ\_E задаём размеры в битах для генерируемых простых целых чисел p, q и числа e.

*const long SZ\_PRIME = 512;*

*const long SZ\_E = 256;*

В теле основной программы в первую очередь инициализируем генератор псевдослучайных чисел функцией SetSeed из библиотеки NTL и инициализируем все необходимые переменные.

*SetSeed(to\_ZZ((double)time(NULL)));*

*ZZ p, q, n, eulerN, e, d, plainText, cipherText, decryptedCipherText;*

string plainText, decryptedCipherText;

Затем последовательно вызовем функции GenPrimes (генерация простых чисел p и q, вычисление их произведения и функции Эйлера), GenKeys (генерация e и d).

*GenPrimes(p, q, n, eulerN, SZ\_PRIME);*

*GenKeys(e, d, eulerN, SZ\_E);*

Сохраним открытый и закрытый ключи в текстовые файлы.

*SaveKey(e, d, n);*

Теперь всё готово для шифрования сообщения. Вводится текст, состоящий из символов, цифр и знаков, который надо зашифровать (plainText). Он кодируется функцией Encryption, в результате чего получается зашифрованный текст (cipherText). Расшифровываем шифротекст с помощью функции Decryption и получаем исходное сообщение.

*cipherText = Encryption(plainText, e, n);*

*decryptedCipherText = Decryption(cipherText, d, n);*

**Результаты работы программы**

Здесь приведены результаты работы программы при размере простых чисел p и q в 512 бит. Программа работает и с числами большего размера, но для её работы за разумное время необходимо оборудование с большей производительностью вычислений. Простое число размера 512 бит генерируется в течение 5 секунд на процессоре Intel Core i5-5200U.



**Список использованной литературы**

1. Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В. - Основы криптографии: Учебное пособие. М.: Гелиос АРВ, 2005.

2. Stinson D. R. Cryptography. Theory and practice. 3 ed. Chapman & Hall/CRC. 2006.

3. Виноградов И. М. - Основы теории чисел: Учебное пособие. СПб.: Издательство «Лань», 2006.

4. Бабаш А. В. – Криптографические методы защиты информации. Т. 1: Учеб.-метод. Пособие. М.: РИОР: ИНФРА-М, 2013.

5. Шнайер Б. – Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2002.