# лабораторная работа №1 “Реализация шифрования методом RSA. Оптимизации и закладки”

## Цель работы

Целью работы является знакомство с принципами работы алгоритма RSA для шифрования и подписи сообщений.

## Основные сведения

В 1978 г. Рон Райвест (Ron Rivest), Ади Шамир (Adi Shamir) и Лен Адлеман (Len Adleman) предложили криптографический алгоритм с открытым ключом. Основная идея заключается в том, чтобы использовать ключи парами, состоящими из ключа зашифрования и ключа расшифрования, которые невозможно вычислить один из другого.

**Процесс шифрования**.

Исходный текст должен быть переведен в числовую форму, этот метод считается известным. В результате этого текст представляется в виде одного большого числа. Затем полученное число разбивается на части (блоки) так, чтобы каждая из них была числом в промежутке [0, *N –* 1]. Процесс шифрования одинаков для каждого блока. Поэтому мы можем считать, что блок исходного текста представлен числом *x*, .

Каждый абонент вырабатывает свою пару ключей. Для этого он генерирует два больших простых числа *p* и *q*, вычисляет произведение . Затем он вырабатывает случайное число *e*, взаимно простое со значением функции Эйлера от числа *N*,  и находит число *d* из условия *e∙d* = 1(mod φ(*N*)). Так как , то такое число *d* существует и оно единственно. Пару (*N*, *e*) он объявляет открытым ключом и помещает в открытый доступ. Пара (*N*, *d*) является секретным ключом. Для расшифрования достаточно знать секретный ключ. Числа *p*, *q*,  в дальнейшем не нужны, поэтому их нужно уничтожить.

Пользователь *A*, отправляющий сообщение *x* абоненту *B*, выбирает из открытого каталога пару (*N*, *e*) абонента *B* и вычисляет шифрованное сообщение . Чтобы получить исходный текст, абонент B вычисляет . Так как , т. е. , где *k* – целое, то применяя теорему Эйлера, получим: следующее соотношение: .

**О вычислениях**

Как шифрование, так и расшифрование в RSA предполагают использование операции возведения целого числа в целую степень по модулю *N*. Если возведение в степень выполнять непосредственно с целыми числами и только потом проводить сравнение по модулю *N*, то промежуточные значения окажутся огромными. Здесь можно воспользоваться свойствами арифметики в классах вычетов . Таким образом, можно рассмотреть промежуточные результаты по модулю *N*. Это делает вычисления практически выполнимыми.

**О стойкости RSA**

Безопасность алгоритма RSA основана на трудоемкости разложения на множители больших чисел. Современное состояние технических средств разложения на множители таково, что число, содержащее 212 десятичных знака (RSA-704), факторизовано в 2012 г. Следовательно, выбираемое *N* должно быть больше. Большинство общепринятых алгоритмов вычисления простых чисел p и q носят вероятностный характер.

**О выборе чисел *p* и *q***

Для работы алгоритма RSA нужны простые числа. Наиболее приемлемым является генерация случайных чисел и последующая проверка их на простоту. Существуют вероятностные тесты, определяющие с заданной степенью достоверности факт простоты числа. Возникает вопрос, что произойдет, если числа окажутся составными? Можно свести вероятность такого события до приемлемого минимума, используя тесты на простоту. Кроме того, если такое событие произойдет, это будет быстро обнаружено — шифрование и расшифрование не будут работать.

Кроме разрядности *p* и *q*, к ним предъявляются следующие дополнительные требования:

– числа не должны содержаться в списках известных больших простых чисел;

– они не должны быть близкими, так как иначе можно воспользоваться для факторизации *N* методом Ферма и решить уравнение .

– в алгоритме RSA всегда есть эквивалентные по расшифрованию показатели степеней, например *d* и . При этом эквивалентных решений тем больше, чем больше (*p –* 1, *q* – 1). В лучшем случае (*p –* 1, *q –* 1) = 2, *p* = 2*t* + 1, *q* = 2*s* + 1, где *s*, *t* – нечетные числа с условием (*s*, *t*) = 1.

Чтобы исключить возможность применения методов факторизации накладывают следующее ограничение: числа *p –* 1, *p* + 1, *q –* 1, *q* + 1 не должны разлагаться в произведение маленьких простых множителей, должны содержать в качестве сомножителя хотя бы одно большое простое число. В 1978 г. Райвест сформулировал наиболее сильные требования. Числа  должны быть простыми, причем числа *p*1 – 1 и *q*1 – 1 не должны разлагаться в произведение маленьких простых.

**О выборе параметров *e* и *d***

Рассмотрим вопрос о выборе экспонент шифрования и расшифрования. Так как значения *е* и *d* определяют время зашифрования и расшифрования, то можно назвать ряд ситуаций, в которых желательно иметь малое значение *е* и *d.* Например, при использовании системы RSA при защите электронных платежей с применением кредитных карточек естественным является требование использования небольших значений экспоненты *d* у владельца карточки и большого значения экспоненты *e* у центрального компьютера.

Однако выбор малых параметров *е* или *d* представляется небезопасным по ряду соображений. Если малым является секретный параметр *d*, то можно применить метод перебора малых значений до получения искомого числа *d*. А если малым является параметр *е*, то достаточно большое число открытых сообщений, удовлетворяющих неравенству  будут зашифровываться простым возведением в степень  и поэтому их можно найти путем извлечения корня степени *е*.

Другая аналогичная ситуация может сложиться, когда у нескольких абонентов используется одинаковая экспонента *е*. Тогда становится возможна атака на основе китайской теоремы об остатках.

**Подготовка текста к шифрованию**

Для выполнения шифрования нужно каким-либо способом представить текст сообщения в виде упорядоченного набора чисел по модулю *N*. Это еще не процесс шифрования, а только подготовка к нему. Разбиение числа на блоки можно произвести различными способами. При этом *промежуточные* результаты зависят от способа разбиения, однако *конечный* результат – не зависит.

**Оптимизации**.

При расшифровании или подписывании сообщения в алгоритме RSA показатель вычисляемой степени будет довольно большим числом, поэтому во многих популярных криптографических библиотеках (OpenSSL, Java и .NET) используется алгоритм, основанный на Китайской теореме об остатках, сокращающий количество операций. Для этого предварительно вычисляются следующие значения (которые хранятся как часть закрытого ключа):

p и q,

d_P = d\text{ (mod }p - 1\text{)},

d_Q = d\text{ (mod }q - 1\text{)} и

q_\text{inv} = q^{-1}\text{ (mod }p\text{)}.

Эти значения позволят на принимающей стороне более эффективно вычислять m = cd (mod pq) используя:

m_1 = c^{d_P}\text{ (mod }p\text{)}

m_2 = c^{d_Q}\text{ (mod }q\text{)}

h = q_\text{inv}(m_1 - m_2)\text{ (mod }p\text{)}, а, если m_1 < m_2, то в некоторых бибилиотеках *h* вычисляется как q_\text{inv}((m_1 + \left\lceil q/p \right\rceil p) - m_2)\text{ (mod }p\text{)}

m = m_2 + hq\,

Это гораздо более эффективно, нежели вычисление m ≡ cd (mod pq), даже если требуется всего 2 возведения в степень по модулю. Причина этого кроется в том, что в случае применения Китайской теоремы об остатках вычисления будут проводиться над на порядок меньшей экспонентой и с меньшим модулем.

**Закладки в RSA**.

Ознакомиться со статьей [«Встраиваем бэкдор в публичный ключ RSA»](http://habrahabr.ru/post/248269/)

## Задание

Необходимо реализовать генерацию ключей, операции зашифрования и расшифрования. Выполнить оптимизацию расшифрования с помощью китайской теоремы об остатках. Оценить производительность классической реализации в сравнении с использованием Китайской теоремы об остатках на различных параметрах *e* и *d*. Графически отобразить зависимость скорости работы от размера ключа.

По статье [«Встраиваем бэкдор в публичный ключ RSA»](http://habrahabr.ru/post/248269/) выполнить внедрение закладки в генерируемые ключи. Оценить накладные расходы на внедрение закладки.

## Порядок выполнения

1. Изучить основы шифрования данных с помощью RSA.
2. Разработать алгоритм и написать программу, обеспечивающую генерацию пары закрытый ключ/открытый ключ, ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.
3. Выполнить оптимизацию с использованием Китайской теоремы об остатках
4. Практически оценить производительность, построить график зависимости скорости работы от размера ключа.
5. Реализовать внедрение закладки в публичный ключ по методу из статьи [«Встраиваем бэкдор в публичный ключ RSA»](http://habrahabr.ru/post/248269/).
6. Практически оценить накладные расходы на внедрение закладки.

## Содержание отчета

1. Цель работы.
2. Задание.
3. Краткая теория.
4. Выдержки из текста программы, реализующие критичные для работы шифрования участки.
5. Пример открытого текста и соответствующей ему шифрограммы.
6. Графики производительности.
7. Выводы по работе.