МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«19» 04 2021 г.

АЛГОРИТМ ШИФРОВАНИЯ RSA

Методические указания по выполнению практических работ
для студентов направления подготовки (специальности)
02.03.03 математическое обеспечение и администрирование
информационных систем

Курск 2021

2

УДК 004.56.5(076.5)
Составители: А.Л. Марухленко, А.Л. Ханис

Рецензент
Кандидат технических наук, доцент кафедры
информационной безопасности А.Г. Спеваков

Алгоритм шифрования RSA : методические указания по выполнению практических работ студентов всех форм обучения / Юго-Зап. гос. ун-т; сост.: А.Л. Ханис. - Курск, 2021. - 19 с. Библиогр.: с. 19.

Содержат сведения по вопросам шифрования и расшифрования RSA. Указывается порядок выполнения практической работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания соответствуют требованиям программы по направлению подготовки бакалавров: математическое обеспечение и администрирование информационных систем.

Предназначены для студентов направления подготовки бакалавров 02.03.03.

Текст печатается в авторской редакции

Подписано в печать 19.04.21. Формат 60x84 1/16.
Усл.печ. л. 1,1 Уч.-изд. л. 1. Тираж 30 экз. Заказ 323 Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

1. ЦЕЛЬ РАБОТЫ

Получение навыков создания зашифрованного сообщения при помощи алгоритма шифрования RSA.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Основной смысл способа с открытым ключом состоит в том, чтобы применить пару ключей, состоящих из ключа шифрования и ключа расшифрования. В данном примере первичное сообщение, которое шифруется открытым ключом, допускается расшифровать только с использованием закрытого ключа. Открытый и закрытый ключи объединены математическими зависимостями, из этого следует что найти пару этих ключей не за короткое время не представляется возможным. В общем случае шифрование открытым и закрытым ключом можно подразумевать как чемодан, для открытие которого используется два ключа: одним - это закрыть, другим – открыть. Когда чемодан закрыли первым ключом, открыть его может будет только вторым, если закроем вторым, для открытия потребуется первый

Алгоритмы, построенные на представленной концепции, являются общедоступными, а, следовательно, необходимости в секретных каналах связи нет. Проанализируем схему взаимодействия между двумя адресатами с использованием данных алгоритмов:

1.Каждый адресат формирует пару ключей: один для шифрования, другой для расшифрования.

2.Каждый адресат объявляет свой ключ шифрования, выставляя его в открытый доступ. Второй ключ сохраняется в секрете.

3. Если адресат А думает послать сообщение адресату В, он шифрует свое сообщение открытым ключом адресата В.

4. Когда адресат В получает сообщение, он расшифровывает его при помощи своего секретного ключа. Другой адресат не сможет расшифровать сообщение, так как секретным ключом владеет только В.

Начало асимметричным шифрам было положено в работе «Новые направления в современной криптографии» Уитфилда Диффии Мартина Хеллмана, опубликованной в 1976 году.

Находясь под влиянием работы Ральфа Меркле (англ. RalphMerkle) о распространении открытого ключа, они предложили метод получения секретных ключей, используя открытый канал. Этот метод экспоненциального обмена ключей, который стал известен как обмен ключами Диффи—Хеллмана, был первым опубликованным практичным методом для установления разделения секретного ключа между заверенными пользователями канала. В 2002 годуХеллман предложил называть данный алгоритм «Диффи—Хеллмана—Меркле», признавая вклад Меркле в изобретение криптографии с открытым ключом. Эта же схема была разработана Малькольмом Вильямсоном в 1970-х, но держалась в секрете до 1997 года. Метод Меркле по распространению открытого ключа был изобретён в 1974и опубликован в 1978 году, его также называют загадкой Меркле.В 1977 год учёными Рональдом Ривестом, Ади Шамироми Леонардом Адлеманом из Массачусетского технологического института был разработан алгоритм шифрования, основанный на проблеме о разложении на множители. Система была названа по первым буквам их фамилий (RSA—Rivest, Shamir, Adleman). Эта же система была изобретена в 1973 году Клиффордом Коксом (англ. CliffordCocks), работавшим в центре правительственной связи (GCHQ), но эта работа хранилась лишь во внутренних документах центра, поэтому о её существовании было неизвестно до 1977 года. RSA стал первым алгоритмом, пригодным и для шифрования, и для цифровой подписи. Кроме RSA существуют другие ассиметричные шифры: DSA, Elgamal, Diffie-Hellman, ECDSA, ГОСТ Р 34.10-2001, Rabin, Luc, McEliece, Williams System и др.

Процесс шифрования заключается в следующем:

1) исходное сообщение, которое мы ходит зашифровать и отправить, необходимо перевести в некую числовую форму;

2) данный набор символов собираем в одно большое число.

3) данное число разбиваем на блоки так, чтобы каждый из блоков был числом в промежутке [0, N – 1].

4) следуя тому что шифрование одинаково для каждого блока, то мы полагаем, что блок первичного сообщения представлен числом x, где $0 < x < N - 1$.

Любой из пользователей, которые собираются обменяться

информацией, формирует пару ключей. Для этого пользователю необходимо подобрать два больших простых числа «p» и «q» и вычислив число «N» , равно произведению чисел p•q. После чего подбирается такое случайное число «e», которое будет взаимно простым со значением функции Эйлера от числа «N», и находит число «d» из условия . Так как число «e» взаимно простое со значение функции Эйлера, то данное число «d» существует, и оно единственно. Пара (N, e) объявляется открытым ключом и размещается в открытый доступ публичной сети. Пара (N, d) является секретным ключом. Для расшифрования нужно обладать секретным ключом [12].

Адресат A, посылающий сообщение X адресату B, находит из открытого каталога пару (N, e) адресат B и рассчитывает шифрованное сообщение

$y = x^e \bmod(N)$. Чтобы получить исходный текст, абонент B рассчитывает $x = y^d \bmod(N)$.

Пример. Пусть p = 7, q = 13. Тогда N = 7*13 = 91, $\varphi(N)$ = 72. Выбираем значение e: e < 72, (e, 72) = 1. Пусть в нашем случае e = 5. Находим d из выражения $d \cdot e \equiv 1(\bmod \varphi(N))$, подставляя выведенное значения. Получаем d = 29. Открытый ключ (91,5), закрытый ключ (91,29). Пусть x = 32. Для шифрования шифрованное сообщение переведенное в численное выражение, в нашем случае 32, возводим в степень «e», в данном случае 5, и берем по модулю числа N, в нашем случае 91, тогда имеем шифрованное сообщение равное 2. Итак,

$y = 32^5 \bmod(91) \triangleright y = 2$, а расшифрование $x = 2^{29} \bmod(91) \triangleright x = 32$.

Алгоритм Евклида нахождения наибольшего общего делителя (число, которое делит без остатка два числа и делится само без остатка на любой другой делитель данных двух чисел) заключается в выполнении ряда этапов: .

1.Большее число делим на меньшее.

2.Если делится без остатка, то меньшее НОД (следует выйти из цикла).

3.Если есть остаток, то большее число заменяем на остаток от деления.

4.Переходим к пункту 1.

Пример: НОД для 30 и 18.30/18 = 1 (остаток 12)18/12 = 1 (остаток 6)12/6 = 2 (остаток 0). Конец: НОД –это делитель. НОД (30, 18) = 6/

Алгоритм решения уравнения вида ax+by = 1 предполагает выполнение следующих итераций:

1.Определим матрицу E = 1 0 / 0 1

2.Вычислим r-остаток от деления числа a на b, a=bq+r, 0 <= r < b.

3.Если r=0, то второй столбец матрицы E даёт вектор (x, y)решений уравнения.

4.Если r не равно нулю, то заменим матрицу E матрицей E* 0 1 / 1 -q

5.Заменим пару чисел (a,b)на (b,r)и перейдем к шагу 2.

Пример умножения матриц: A=1 2/ 3 4 , B= 5 6 / 7 8
c1,1 = a1,1b1,1+a1,2b2,1 = 1*5+2*7= 5+14= 19
c1,2 = a1,1b1,2+a1,2b2,2 = 1*6+2*8= 6+16= 22
c2,1 = a2,1b1,1+a2,2b2,1 = 3*5+4*7= 15+28= 43
c2,2= a2,1b1,2+a2,2b2,2 = 3*6+4*8= 18+32= 50
Результирующая матрица |C|=19 22 / 43 50.

Часто возникает задача проверки натурального числа на простоту. При этом имеются вероятностные и детерминированные методы проверки. Здесь рассматриваются только детерминированные алгоритмы, дающие 100% ответ на вопрос о простоте. Хорошо известно такое утверждение: если натуральное число n>1 не делится ни на одно простое число, не превосходящее $\sqrt{n}$, то оно простое. В связи с этим получается самый простой способ проверки на простоту алгоритм В данном алгоритме из множества (2,3… $\sqrt{n}$) отброшено 50% четных чисел, так как если число a не делится на 2, то нет смыла делить его на 4, 6 и т.д. Данный метод можно усовершенствовать и отбросить из множества (2,3… $\sqrt{n}$) больше чисел. Для этого выбирается некоторое число m, равное произведению простых чисел без степеней и рассматриваются только те элементы множества (2,3… $\sqrt{n}$), которые взаимно просты с m. Например, если m = 6 = 2*3, то из

этого множества отбрасывается 66% элементов (ненужных проверок). В этом случае алгоритм будет быстрее предыдущего при больших n. Если m = 30 = 2*3*5, то такой алгоритм будет еще быстрее и отбрасывает уже 74% лишних элементов.

Данный пример носит лишь информативный характер работы представленного криптографического алгоритма, на практике применение столь малых чисел чревато последствиями, а именно раскрытие шифруемого сообщения, ниже будут представлены советы по поддержанию криптостойкости данного алгоритма и приведены несколько видов атак на него.

Процессы шифрования и расшифрования в криптографическом алгоритме RSA основываются на применении операций возведения целых чисел в целую степень по модулю N. В том случае если возведение в степень выполнять непосредственно с целыми числами и далее провести сравнение по модулю N, то промежуточные значения будут большими. Само собой , не нужно проводить многократное количество умножений, а потом брать остаток от деления числа из полученных таким путем цифр. Остаток от деления берется после каждого умножения. Следовательно, при перемножении двух чисел, состоящих из Z бит потребуется 2 * Z-битное число, которое в свою очередь делится на модуль и выводиться остаток, опять же состоящий из Z бит.

Проблема данного алгоритма является ь оценена как O(lnT), где T - модуль, по которому осуществляется умножение. Регистрация O(lnT) ведет к тому что, что для реализации алгоритма потребуется порядка lnT операций умножения по модулю. Например, если число имеет разрядность 1024 бит , то умножение по модулю необходимо будет провести порядка lnT = ln 21024 = 710 раз, что относительно немного.

Основанием считать криптографический алгоритм RSA безопасным основано на трудоемкости разложения числа «N» на множители больших простых чисел (факторизации). На сегодняшний день в качестве чисел «e» и «d» рекомендуется подбирать такого рода числа, длина которых будет не меньше 768 бит. Для того что бы найти ключ данной длины нужно будет примерно год времени. Ключ в 1024 бит является достаточно надежным для обычных целей шифрования. Для повышения

криптостойкости предлагается подбирать числа не меньше 2048 бит.

Операции с такими большими числами требуют определенных алгоритмов и структур данных. Главные вопросы данной области: как хранить эти числа, как их складывать, умножать, делить, брать остаток от деления, возводить в большую степень по модулю целого числа. Большая часть общепринятых алгоритмов вычисления простых чисел p и q носят вероятностный характер.

В целях взаимодействий с криптографическими алгоритмами RSA нужны простые числа, к тому же различные, иначе задача факторизации сводится к вычислению корня в целых числах,в сою очередь которое выполняется за считанные секунды. Существует несколько способов способствующих подбору простых чисел с большой разрядностью Первый способ – это деление предполагаемого простого числа на все числа меньшие его, но он не работоспособен уже с 32-битными числами , да и разрядность слишком мала. Следующий способ заключается в подборе с помощью вероятностных тестов, которые определяют с заданной степенью достоверности факт простоты числа. В том случае, если выбранные числа будут составными, то данная проблема будет мгновенно обнаружена, так как процесс шифрования и расшифрования не будут работать.

Помимо разрядности чисел «p» и «q», для них существуют следующие дополнительные требования:

- данные числа не должны находиться в списках известных больших простых чисел;
- данные числа не должны быть близкими
- в алгоритме RSA всегда есть эквивалентные по расшифрованию показатели степеней, например d и d' = d+(p–1, q–1). При этом эквивалентных решений тем больше, чем больше (p – 1, q – 1). В лучшем случае (p – 1, q – 1) = 2, p = 2t + 1, q = 2s + 1, где s, t – нечетные числа с условием (s, t) = 1.

Чтобы исключить возможность использования 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.

Следующим немаловажным предоставляется выбор числовых значений ключей шифрования и расшифрования. Подбор небольших значений «e» или «d» является небезопасным по нескольким причинам. В случае когда значение секретного состояния «d» достаточно мало, то следовательно можно будет применить метод перебора малых значений до получения первичного числа «d». Если числовое значение характеристики «e» мало является параметр e, то достаточно большое число открытых сообщений, удовлетворяющих неравенству $x < \sqrt[e]{N}$, будут зашифровываться простым возведением в степень $y = x^e$ и следовательно их можно определить если извлечь корень в степени e [12].

Иная похожая ситуация может получиться, когда у многих пользователей используется одна и таже экспонента e. Следовательно возможна атака на основе китайской теоремы об остатках. В том случае, когда используется малую характеристику e, нужно пользоваться добавками в сообщениях, в роли которых могут быть включены в конец сообщения случайные вектора или же время отправки.

Нужно также иметь в виду, что ключи e и d равноправны, т.е. сообщение можно шифровать как ключом e, так и ключом d, при этом расшифровка должна быть произведена с помощью другого ключа.

Для расшифрования необходимо по известным N, e и шифрованному сообщению y найти такое x, что $y = x^e \bmod(N)$.

Один из возможных способов решения представленной задачи, является нахождение таких пары чисел, что: имеется набор пар $\{(x_1, y_1)...(x_i, y_i)\}$ с условием, что $x_i^e = y_i \bmod(N)$, $1 < y < N$, пара чисел y и N взаимнопростые. Если каким–либо образом удалось представить y в виде $y = y_1^{s_1} \cdot ... \cdot y_m^{s_m} \bmod(N)$ с целочисленными s_m, то $x = x_1^{s_1} \cdot ... \cdot x_m^{s_m}$ будет решением задачи, вида $y = x^e \bmod(N)$ [12].

Пример. Известны параметры открытого ключа $N = 31459, e = 5$ и набор пар соответствующих друг другу исходных и шифрованных сообщений: $(23;18707), (755;26871), (631;6384)$. Требуется расшифровать шифрованное сообщение, переведенное в числовую последовательность, вида y = 11 638. Для этого представим y в виде

$y = 18707^{-1} \cdot 26871^{3} \cdot 6384^{-2} = 11638$. Далее не составит труда вычислить исходное сообщение $x = 23^{-1} \cdot 755^{3} \cdot 631^{-2} = 28260$.

Стоит отметить, что представленный метод не менее труден, чем поиск алгоритма решения задачи вида $y = x^{e} \bmod(N)$

Так же возможны ситуации, когда пользователь выбирает слишком малое значение параметра открытого ключа e (3,5,7 и др.) Предположим, что пользователь решил отправить одно и то же сообщение трем другим пользователям. Затем пользователь шифрует исходное сообщение с открытыми ключами (e, N_1), (e N_2), (e, N_3). Злоумышленник перехватывает все три сообщения и составляет систему уравнений вида:

$$\begin{cases} y = y_1 \bmod(N_1) \\ y = y_2 \bmod(N_2) \\ y = y_3 \bmod(N_3) \end{cases}$$

Поскольку $o \le y \le N \quad x^{e} < N$, то злоумышленник вычислив корень степени e из c, найден исходное сообщение x.

Основными атаками на алгоритм RSA считаются:

1. Атака повторным шифрованием. Известны следующие характеристики открытого ключа N =84517, e = 397 и шифрованное им сообщение y = 8646. Нужно найти первичны текст x. Возведем y в степень e и получим y_2 = 37043. Данную операцию будем повторять до тех пор, пока не получим $y_n = y \cdot y_{n-1}$, являющимся первичным сообщением,y_3 = 5569, y_4 = 61833, y_5 = 83891, y_6 = 16137, y_7 = 8646=y – является решением задачи вида $y = x^{e} \bmod(N)$, а, так же, первичным сообщением x [11]. Оценка метода повторного шифрования показывает то что нужно соблюдать требования при выборе p и q для осуществления большей стойкости. В этом примере d = 82225. Из за того что был сделан неудачный выбор криптосистемы, этот выбор привел к тому , что атака методом повторного шифрования дала результат почти сразу, тогда как нахождение d потребовало бы на порядок больших вычислений.

2. Атака на основе Китайской теоремы об остатках. Системы шифрования с открытыми ключами работают медленно. Для увеличения скорости шифрования RSA пользуются малой экспонентой шифрования. Когда мы выбираем число e

минимальным или таким, чтобы в его двоичной записи было мало единиц, то процесс шифрования можно в значительной степени ускорить. Например, выбрав e = 3, мы можем использовать шифрование с помощью одного возведения в квадрат по модулю N и одного перемножения. Выбрав e = 216–1 = 65 537 – число, двоичная запись которого содержит только две единицы, мы сможем реализовать шифрование с помощью шестнадцати возведений в квадрат по модулю N и одного перемножения. Когда экспонента e выбирается случайно, то реализация шифрования по алгоритму RSA потребует S возведений в квадрат по модулю N и в среднем вдвоем меньше умножений по тому же модулю, где S – длина двоичной записи числа N. Вместе с тем выбор небольшой экспоненты e может привести к плачевным последствиям [11].

Так же при выборе малой экспоненты расшифрования d также нежелателен в связи с возможностью определения d простым перебором. Известно также что если $d < \sqrt[4]{N}$, то экспоненту d легко найти, используя непрерывные дроби.

3. Бесключевое чтение. Пусть два абонента выберут одинаковый модуль N и разные экспоненты e1 и e2. Когда один абонент посылает им некое циркулярное сообщение x, то криптоаналитик противника сможет получить в свое распоряжение два шифрованных текста $y_1 = x^{e_1} \bmod(N)$ и $y_2 = x^{e_2} \bmod(N)$. В данной ситуации криптоаналитик может получить первичное сообщение, использовав расширенный алгоритм Евклида, находим r, s такие, что $r \cdot e_1 + s \cdot e_2 = 1$. В результате получаем $y_1^r y_2^s = x^{re_1 + s \cdot e_2} = x \bmod(N)$

Пример. Два абонента применяют одинаковый модуль N = 137759, но различные взаимно простые экспоненты e1 = 191 и e2 = 233. Абоненты получили зашифрованное сообщения y1 = 60197 и y2 = 63656, в котором содержится одно и то же сообщение. Найдем первичное сообщение методом бесключевого чтения. Так как e1 и e2 взаимно просты, то найдем такие r и s, что $r \cdot e_1 + s \cdot e_2 = 1$. При помощи расширенного алгоритма Евклида находим r = 61, s = –50. Первичное сообщение $x = y_1^r \cdot y_2^s = 60197^{61} \cdot 63656^{-50} = 1234$.

Как мы видим выбор характеристик криптосистемы является важной задачей. Характеристики нужно выбирать в строгом соответствии с требованиями. Существующими в настоящими

время методами атака на алгоритм и/или криптосистему возможна лишь при неправильном выборе характеристик.

И так, система RSA, в основном, применяется для защиты ПО и в схемах цифровой подписи. В свою очередь она так же используется в открытой системе шифрования и иных системах шифрования в сочетании с симметричными алгоритмами. Но из-за низкой скорости шифрования , сообщения обычно шифруют с помощью более производительных симметричных алгоритмов со случайным сеансовым ключом , то с помощью RSA шифруют лишь этот ключ, следовательно реализуется гибридная криптосистема. Данный механизм имеет определенные уязвимости ввиду необходимости использовать криптографически стойкий генератор псевдослучайных чисел для формирования случайного сеансового ключа симметричного шифрования. Однако, злоумышленник, перехвативший симметричный ключ, вероятнее всего, начнем атаку на перехваченное сообщение, что не приведет его к успеху, так как коллаборация двух видов криптографических преобразований (на базе асимметричного и симметричного шифрования) значительно увеличивает криптостойкость.

В связи с тем, что алгорим RSA предполагает возведение в степень числа открытого ключа значений исходных блоков, целесообразно упрощение вычислений за счет применения операции mod на каждой итерации умножения. Для наглядности приведем код функции ABC, результат которой остаток от деления на число C величины A^B и функции проверки простоты числа (isSimple). Реализация на базе HTML+JavaScript без использования дополнительных расширений и зависимостей.

```
<!DOCTYPE html>
<script>
    function ABC() {
        a = document.getElementById("A").value;
        b = document.getElementById("B").value;
        c = document.getElementById("C").value;
        r = a % c;
        for (i = 2; i <= b; i++) {
            r = (r * a) % c;
        }
        document.getElementById("R").innerHTML =
        ' = '+a+'^'+b+' mod ' + c + ' = ' + r;
        document.getElementById("A").style.color = IsSimple(a);
    }
    function IsSimple(a) {
        b = "green";
        f = true;
        for (i = 2; i < Math.sqrt(a) && f; i++) {
            if (!(a % i)) {
                f = false;
                b = "red";
            }
        }
        return b;
    }
</script>
<body>
    A=<input id="A" type="number" value="123">
    B=<input id="B" type="number" value="456">
    C=<input id="C" type="number" value="789"><br><br>
    <button onclick="ABC()">A^B mod C</button><label id="R"></label>
</body>
</html>
```

3. ВЫПОЛНЕНИЕ РАБОТЫ

В рамках практической работы необходимо произвести блочное шифрование методом RSA. В качестве входной информации берется индивидуальный вариант задания в соответствии номером студента по списку.

№	данные	p	q	e
1	аврора	307	70001	2383
2	атриум	439	42451	3529
3	бахилы	653	20873	3533
4	бензин	2803	18397	3539
5	бизнес	2819	9323	3541
6	версия	2897	5531	3617
7	взгляд	2903	3037	3623

8	власть	2909	2683	3631
9	вокзал	2927	2671	3643
10	газель	2939	2663	3659
11	гектар	2953	2659	3673
12	глазок	2957	2659	3677
13	голубь	2963	2647	3691
14	джинсы	2999	2617	3709
15	добыча	3011	2593	3727
16	дружба	3019	2591	3733
17	экватор	3041	2557	3761
18	жалюзи	3049	2551	3767
19	жребий	3061	2549	3779
20	забота	3083	2539	3797
21	звание	3089	2531	3803
22	значок	3109	2521	3821
23	зубило	3119	2503	3823
24	импорт	3121	2477	3847
25	йогурт	3163	2473	3851
26	казино	3169	2459	3863
27	карета	3187	2447	3877
28	квинта	3191	2441	3881
29	кетчуп	3203	2437	3907
30	климат	3221	2411	3919
31	кредит	3259	2389	3943
32	лагерь	3271	2381	3967
33	ледник	3301	2377	3989
34	нокаут	3319	2351	4013
35	огурец	3329	2347	4019
36	павлин	3359	2333	4051
37	пиджак	3373	2297	4091
38	победа	3389	2287	4093
39	радуга	3413	2269	4129
40	регион	3433	2267	4133
41	сапоги	3463	2237	4159
42	сделка	3469	2221	4177
43	сериал	3491	2207	4201

44	собака	3499	2203	4217
45	стирка	3517	2179	4219
46	счастье	3527	2161	5483
47	таймер	4261	2153	8059
48	творец	4523	2141	8089
49	термин	4999	2131	8293
50	топчан	5179	2129	8297
51	трофей	7639	2113	8423
52	уборка	7879	2111	8783
53	угодье	8009	2099	9157
54	улитка	8117	2089	9341
55	утопия	8363	2083	9439
56	фонарь	8543	2069	10099
57	хозяин	9311	2029	65539
58	яблоко	9463	2017	71363
59	чайник	10093	1093	75209
60	шаблон	17467	971	101573
61	шинель	21713	691	105323
62	щетина	42197	569	105529
63	экипаж	70207	353	1000403
64	юбиляр	98467	167	1000457
65	янтарь	8117	65539	65539
66	яхтинг	2111	3499	105529

Порядок выполнения работы:

1. Вычислить закрытый ключ.
2. Определить длину блока.
3. Сформировать десятичное представление исходных блоков.
4. Рассчитать шифр-блоки.
5. Выполнить проверку.
6. Результат шифрования представить в виде 16-битных hex-блоков.
7. Сохранить результат в файл, в котором результирующие блоки в виде цепочки бит на расстоянии N (номер студента по списку).

4. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Рассмотрим решение типового варианта, входные данные: знаток; p = 3083; q = 2539; e = 3797;

Выполнение работы.

1. Вычисляем закрытый ключ.

В связи с тем, что требованием к закрытому числу является выполнение e*d mod f = 1, целесообразно использование расширенного алгоритма Евклида, определяющего поиск решений выражения e*d + x (p-1)*(q-1)=1.

Вычисляем закрытый ключ:

1) a= 7822116 b= 3797, div= 2060 mod= 296
|0 1
|1 -2060
2) a= 3797 b= 296, div= 12 mod= 245
|1 -12
|-2060 24721
3) a= 296 b= 245, div= 1 mod= 51
|-12 13
|24721 -26781
4) a= 245 b= 51, div= 4 mod= 41
|13 -64
|-26781 131845
5) a= 51 b= 41, div= 1 mod= 10
|-64 77
|131845 -158626
6) a= 41 b= 10, div= 4 mod= 1
|77 -372
|-158626 766349
e*d mod f =3797*766349 mod 7822116=1

Проверка:

3797*766349 mod 7822116 = 1 (верное равенство),

закрытый ключ (d, n) = (766349, 7827737).

2. Определяем длину блока.

В соответствии с алгоритмом, длина блока len = [log2n], соответственно

len = [log2 7827737] = 22 бит.

3. Формируем исходные блоки и производим шифрование.

Для формирования десятичного представления исходных блоков по len бит запишем исходные данные последовательностью ASCII-кодов:

знаток = 231; 237; 224; 242; 238; 234;

формируем двоичную последовательность старшими битами вправо:

11100111 10110111 00000111 01001111 01110111 01010111

блочная обработка:

1). ст-> 11100111101101110000001 мл-> 10000011101101111100111 = 2158055

2158055^3797 mod 7827737 = 7056313

2). ст-> 11010011111011101110101 мл-> 10101110111011111001011 = 2866123

2866123^3797 mod 7827737 = 1440910

3). ст-> 01110000000000000000000 мл-> 00000000000000000001110 = 14

14^3797 mod 7827737 = 242127

4. Результат представляем в виде 16-битных блоков в шестнадцатеричной системе счисления.

Учитывая, что для учета каждого выходного блока потребуется len+1 бит, формируем результирующую двоичную последовательность (нормируем по 16 бит):

старшими битами вправо:

1001110111010101 1101011011100010 0111111101010011 1100111000110111 0000000000000000

младшими битами вправо:

1010101110111001 0100011101101011 1100101011111110 1110110001110011 0000000000000000

результат выполнения работы:

ABB9 476B CAFE EC73 0000

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Описать принцип работы алгоритма RSA.

2 Привести блок-схему алгоритма, вычисляющего ABmodNдля чисел большого порядка.

3 Используя расширенный алгоритм Евклида определить закрытый ключ.

4 Какое максимальное число можно учесть используя стандартные целочисленные типы данных в Delphi?

5 Привести блок-схему расширенного алгоритма Евклида.

6 Описать методы оптимизации проверки простоты натуральных чисел.

7 Почему при шифровании размер выходного блокаувеличивается?

8 Для заданных P=_____и Q=_____определить максимальное корректное E.

9 Является ли число A=_____простым?

10 Как умножить два числа, если разрядность их произведения не укладывается в стандартные типы данных?

11 Для заданных E=_____и P*Q=_____определить D.

12 Опишите сильные требования к выбору P, Q, сформулированные Райвестом в 1978 г.

13 Решить задачу ДАННЫЕ=_____, P=_____, Q=_____, E=_____

14 Опишите существующие атаки на алгоритм RSA.

15 Используя алгоритм Евклида найти общий делитель чисел A=___, B=_____

16 Подобрать три варианта Dдля параметров шифрования P=___ Q=____ E=____

17 В чем сложность задачи злоумышленника, обладающего открытым ключом и шифром?

18 Где на сегодняшний день применяется метод RSA?

19 Определите размер выходного файла, если входной файл содержит =_____, открытый ключ E=_____, N=_____

20 Перевести в двоичную систему отрицательное число A=_____

21 Опишите принцип работы ассиметричных систем шифрования, отличных от RSA.

22 Назовите принципиальное отличие бесключевых криптоалгоритмов от одно- и двухключевых.

23 Дайте определение понятиям: «криптограмма», «ключ шифрования», «криптографический протокол».

24 Обеспечивает ли криптографическая система целостность информации?

25 В чем заключается сложность вычислений алгоритма RSA?

СПИСОК ЛИТЕРАТУРЫ

1. Панасенко, С. Алгоритмы шифрования [Текст] / С. Панасенко. Спб: БХВ-Петербург, 2009, 576 стр.

2. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография [Текст] / А.Г. Ростовцев, Е.Б. Маховенко. М: АНО НПО "Профессионал", 2005, 480 стр.

3. Рябко, Б. Я., Фионов, А. Н. Основы современной криптографии для специалистов в информационных технологиях [Текст] / Б. Я. Рябко, А. Н. Фионов. М: Научный мир, 2004, 179 стр.

4. Смарт, Н. Криптография [Текст] / Н. Смарт. М: Техносфера, 2006, 528 стр.