Содержание

1. Задание на курсовую работу

Номер варианта для курсовой работы выбирается согласно таблице 1. В данной таблице N – порядковый номер студента по журналу, No – номер варианта курсовой работы.

Таблица 1. Варианты курсовой работы

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
No	23	10	17	9	30	16	21	2	24	15	3	26	12	7	20
N	16	17	18	19	20	21	22	23	24	25	26	27			
No	27	18	29	28	19	11	4	14	13	1	25	6			

Номеру 7 по журналу соответствует 21 вариант курсовой работы.

Текст задания на курсовую работу:

1. Используя математический пакет Matlab, разработать программу, реализующую алгоритмы шифрования и дешифрования RSA в случае двух абонентов A, B.

Рекомендуемые значения параметров приведены в таблице 2.

Таблица 2. Рекомендуемые значения параметров

P_A	Q_A	P_B	Q_B	d_A	d_B
163	241	193	347	3	3

2. Алгоритмы протестировать с использованием кодировки русского алфавита Z_{44} на примере шифрации/дешифрации своей фамилии имени и отчества (ФИО).

2. Актуальность и предыстория секретных систем связи с открытым ключом

Системы связи можно разделить на две большие категории: симметричные и ассиметричные. Под симметричными системами подразумевается передача ключа шифрования по закрытому каналу связи. Примером такого закрытого канала может служить доверенное лицо – курьер, который доставляет ключ шифрования абонентам. Зашифрованное сообщение передается по открытому каналу связи.

С развитием технологий широкое распространение получили ассиметричные системы связи, в которых ключ шифрования состоит и двух частей: открытой и закрытой. Открытая часть ключа доступна всем абонентам, закрытая только владельцу. Сообщения в таких системах шифруются с помощью определенного алгоритма, который использует закрытую и открытую части ключей. И так как закрытая часть ключа абонента, необходимая для дешифрования сообщения, не передается в канал связи, злоумышленнику не представляется возможным дешифрация сообщения.

Каждая категория систем связи имеет свои преимущества и недостатки.

К преимуществам симметричных систем можно отнести: довольно высокую скорость шифрования и дешифрования, а также простоту реализации. Существенным недостатком таких систем является необходимость в закрытом канале связи для передачи ключей шифрования.

Основным достоинством ассиметричных систем является избавление от дорогостоящего и более сложного в обслуживании закрытого канала связи. Однако данные системы не лишены недостатков. Среди которых снижение скорости шифрования и дешифрования. Данный недостаток обусловлен требованиями к вычислительным мощностям абонентов, но с развитием технологий скорость шифрования и дешифрования

увеличивается. Так же существенным недостатком ассиметричных систем является возможность подмены злоумышленником открытого ключа.

В данной работе показаны этапы разработки ассиметричной системы связи на основе алгоритма RSA. Данный алгоритм был разработан в 1977 году Рональдом Ривестом, Ади Шамиром и Леонардом Адельманом из Массачусетского технологического института. Алгоритм был назван по первым буквам фамилий его создателей.

3. Постановка задачи

В данной работе необходимо разработать систему секретной связи с открытым ключом на основе алгоритма RSA. Данную задачу разобьем на этапы:

3.1. Разработка функции кодирования, которая ставит соответствие между a_i символом исходного сообщения алфавита A_{44} и цифровым кодом данного символа. На выходе процедуры – закодированное сообщение.

3.2. Разработка вспомогательной процедуры, реализующей классический алгоритм Евклида.

3.3. Разработка процедуры, реализующей алгоритм нахождения взаимно простого числа.

3.4. Разработка процедуры, реализующей обобщенный алгоритм Евклида.

3.5. Разработка вспомогательной процедуры, реализующей алгоритм вычисления обратного по модулю числа.

3.6. Разработка вспомогательной процедуры перевода десятичного числа в двоичное.

3.7. Разработка процедуры, реализующей алгоритм эффективного возведения в степень.

3.8. Разработка процедуры генерации ключей.

3.9. Разработка процедуры, реализующей шифрование закодированного сообщения с помощью шифра RSA.

3.10. Разработка процедуры, реализующей дешифрование криптограммы.

3.11. Разработка процедуры декодирования цифрового сообщения в буквенный эквивалент.

4. Описание метода решения

В данном разделе курсовой работы подробно изложено подробное математическое описание приведенных в разделе 3 этапов.

4.1. Алгоритм кодирования

Каждый символ исходного буквенного сообщения a_i заменятся цифровой эквивалент данного символа в алфавите. В данной работе для кодирования и декодирования сообщений используется алфавит Z_{44}. Кодирование осуществляется согласно таблице 3.

Таблица 3. Таблица кодирования символов.

a_i		А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
m_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
	14	15	16	17	18	19	20	21	22	23	24	25	26	27
	Ы	Ь	Э	Ю	Я	!	?	.	,	:	;	(	)	“
	28	29	30	31	32	33	34	35	36	37	38	39	40	41
	”	-												
	42	43												

Входные параметры: $\vec{a} = \{a_1, a_2, ..., a_n\}$, $a_i \in A_{44}$, $i = 1, ..., n$

Выходные параметры: $\vec{m} = \{m_1, m_2, ..., m_n\}$, $m_i \in Z_{44}$, $i = 1, ..., n$

Здесь a_i - буквенный символ сообщения,

m_i - закодированный символ сообщения

В дальнейшем данный алгоритм будем использовать в виде функции:

$$m_i = f_1(a_i),\ i = 1, ..., n \tag{1}$$

4.2. Классический алгоритм Евклида

Классический алгоритм Евклида является рекуррентным и служит для нахождения наибольшего общего делителя (НОД) двух чисел.

Входные параметры: $a, b \in N$

Выходные параметры: $c = НОД(a, b),\ c \in N$

Так как данный алгоритм является рекуррентным, он описывается следующим рекуррентным уравнением:

Начальные условия: $r_0 = a,\ r_1 = b$

$$r_{i+1} = r_{i-1} (\mathrm{mod}\, r_i),\ i = 1, 2, ..., n \tag{2}$$

Условие остановки $n : r_{n+1} = 0$

Выход алгоритма: $c = r_n$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$c = f_2(a, b) \tag{3}$$

4.3. Алгоритм нахождения взаимно простого числа

Два числа называются взаимно простыми, если они не имею ни одного общего делителя кроме 1. В курсовой работе данный алгоритм является вспомогательным и служит для вычисления открытой части ключа.

Входные параметры: $a \in N$

Выходные параметры: $r \in N$ - число, взаимно простое с a

Опишем данный алгоритм системой рекуррентных уравнений:

$$\begin{gathered} \text{Начальное условие: } y_1 = 2 \\ k_i = f_2(y_i, a),\ i = 1,...,n \\ y_{i+1} = y_i + 1,\ i = 1,...,n \\ \text{Условие остановки } n : k_{n+1} = 1 \\ \text{Выход алгоритма: } r = y_n \end{gathered} \tag{4}$$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$r = f_3(a) \tag{5}$$

4.4. Обобщенный алгоритм Евклида

С помощью обобщенного алгоритма Евклида можно найти такие числа $x, y \in N$, что будет выполняться равенство $ax + by = c$, где $c =$ НОД(a, b). В данной работе этот алгоритм является вспомогательным и служит для нахождения обратного по модулю числа.

Входные параметры: $a, b \in N$

Выходные параметры: $x, y \in N$ - коэффициенты перед a и b,

$$c = НОД(a, b),\ c \in N$$

Опишем данный алгоритм системой рекуррентных уравнений:

Определим следующие векторные переменные:

$$\vec{u}_i = (u_{i1}, u_{i2}, u_{i3})$$

$$\vec{v}_i = (v_{i1}, v_{i2}, v_{i3})$$

$$\vec{t}_i = (t_{i1}, t_{i2}, t_{i3})$$

$$\begin{gathered} \text{Начальные условия: } \vec{u}_0 = (a,\ 1,\ 0),\ \vec{v}_0 = (b,\ 0,\ 1) \\ q_i = \left\lfloor \frac{u_{i1}}{v_{i1}} \right\rfloor,\ i = 1,...,n \\ \vec{t}_i = \left(u_{i1} (\mathrm{mod}\, v_{i1}),\ u_{i2} - q_i v_{i2},\ u_{i3} - q_i v_{i3}\right),\ i = 1,...,n \\ \vec{u}_{i+1} = \vec{v}_i,\ i = 1,...,n \\ \vec{v}_{i+1} = \vec{t}_i,\ i = 1,...,n \\ \text{Условие остановки } n : v_{n+1,1} = 0 \\ \text{Выход алгоритма:} \\ c = u_{n1}, \\ x = u_{n2}, \\ y = u_{n3} \end{gathered} \tag{6}$$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$\left(c,\ x,\ y\right) = f_4(a, b) \tag{7}$$

4.5. Алгоритм вычисления обратного по модулю числа

Данный алгоритм основан на обобщенном алгоритме Евклида, описанном в разделе 4.4. и необходим для вычисления закрытого ключа для алгоритма шифрования RSA.

Входные параметры: $c, n \in N$

Выходные параметры: $d \in N$ - число, обратное числу c по модулю n ($dc(\mathrm{mod}\, n) = 1$)

Данный алгоритм описывается системой уравнений:

$$\begin{gathered} \left(\sim,\ \sim,\ y\right) = f_4(n, c) \\ d = y(\mathrm{mod}\, n) \end{gathered} \tag{8}$$

Здесь f_4 - функция, реализующая обобщенный алгоритм Евклида, описанная в разделе 4.4.

В дальнейшем данный алгоритм будем использовать в виде функции:

$$d = f_5(n, c) \tag{9}$$

4.6. Алгоритм перевода десятичного числа в двоичное

Данный алгоритм служит для перевода числа из десятичной системы счисления в двоичную. Любое натуральное десятичное число можно представить в двоичной системе счисления. В двоичной системе счисления число записывается с помощью 0 и 1. Алгоритм перевода десятичного числа в двоичное является вспомогательным и используется в алгоритме эффективного возведения в степень.

Входные параметры: $X \in N$ - целое десятичное число;

Выходные параметры: $\vec{x} = \{x_1, x_2, ..., x_n\},\ x_i \in \{0,1\}$ - двоичное представление числа X,

$n \in N$ - количество разрядов в X

Опишем данный алгоритм системой рекуррентных уравнений:

Начальное условие: $n = \lfloor \log_2 X \rfloor + 1$

$$\begin{gathered} x_{i-1} = \left\lfloor \frac{X - x_n 2^{-1}}{2^{i-2}} \right\rfloor,\ i = 1, ..., n \\ x_i = \left\lfloor \frac{X}{2^{i-1}} \right\rfloor,\ i = 1, ..., n \end{gathered} \tag{10}$$

Выход алгоритма: $\vec{x} = \{x_1, x_2, ..., x_n\}; n$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$(\vec{x}, n) = f_6(\mathrm{X}) \tag{11}$$

4.7. Алгоритм эффективного возведения в степень по модулю

Возведение больших чисел в степень по модулю является сложной задачей и зачастую результат оказывается неверным. Что бы избежать этих неприятностей используем алгоритм эффективного возведения в степень по модулю. Для получения правильного результата показатель степени x представляется в двоичной системе счисления, с помощью алгоритма описанного в разделе 4.6. В данном алгоритме биты показателя степени просматриваются от младшего к старшему, то есть справа-налево. Алгоритм возведения в степень по модулю используется при шифровании закодированного сообщения в алгоритме RSA.

Входные параметры: $a, x, p \in N$

Выходные параметры: $y \in N$

Здесь y - число a, возведенное в степень x по модулю p

$$y = a^x (\mathrm{mod}\ p)$$

Опишем данный алгоритм системой рекуррентных уравнений:

$$(x', n) = f_6(x)$$

Начальные условия: $y_1 = 1,\ s_1 = a$

$$y_{i+1} = \begin{cases} y_i s_i (\text{mod } p), & \textit{при } x_i = 1 \\ s_i^{\,2} (\text{mod } p), & \textit{при } x \neq 1 \end{cases},\ i = 1,...,n \qquad (12)$$

$$s_{i+1} = s_i^{\,2} (\text{mod } p),\ i = 1,...,n$$

Выход алгоритма: $y = y_{n+1}$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$y = f_7(a, x, p) \qquad (13)$$

4.8. Алгоритм генерации ключей

Алгоритм RSA подразумевает использование ассиметричной системы закрытой связи, а, следовательно, возникает необходимость в генерации пары ключей (открытый и закрытый) для каждого абонента. Открытый ключ передается в открытый канал связи и известен всем пользователям, а закрытый держится в секрете. Открытый ключ выбирается как случайное число, удовлетворяющее определенным условиям.

Входные параметры: $K_{\min}, K_{\max}, q \in N$

Выходные параметры: $p \in N$

Здесь $K_{\min}, K_{\max}$ - отрезок, на котором происходит поиск открытого ключа,

q - число, с которым найденный ключ должен быть взаимно простым,

p - найденный ключ

Опишем данный алгоритм системой уравнений:

Начальные условия: $z_0 = K_{\min}$

$$z_i = K_{\min} + \left\lfloor (K_{\max} - K_{\min})\, rand(i) \right\rfloor,\ i = 1,...,n \qquad (14)$$

Условие остановки: $n : (z_n \neq z_{n-1}) \& (f_2(z_n, q) = 1)$

Выход алгоритма: $p = z_n$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$p = f_8(K_{\min}, K_{\max}, q) \qquad (15)$$

4.9. Алгоритм шифрования

Алгоритм шифрования RSA заключается в возведении в степень каждого закодированного символа исходного сообщения в степень d по модулю n. “Хитрость” данного алгоритма заключается в формировании ключей. Вычисление $n = pq,\ p, q \in P$ является довольно легкой задачей, а вычислить p и q, зная только число n, является довольно сложной задачей. Это называется задачей факторизации. Таким образом злоумышленнику необходимо потратить довольно много времени чтобы узнать закрытые ключи для дешифрования сообщений, так как закрытые ключи абонентов формируются с помощью чисел p и q.

Входные параметры: $\vec{m} = \{m_1, m_2, ..., m_n\}$ - закодированное сообщение,

$d, n \in N$ - открытые ключи

Выходные параметры: $\vec{c} = \{c_1, c_2, ..., c_n\}$ - зашифрованное сообщение

Алгоритм шифрования описывается уравнениями:

от А - к Б

$$\begin{aligned} n_Б &= p_Б q_Б, && \text{(а)} \\ \varphi_Б &= (p_Б - 1)(q_Б - 1), && \text{(б)} \\ d_Б &= f_8(0, \varphi_Б, \varphi_Б), && \text{(в)} \\ c_{iA} &= f_7(m_{iA}, d_Б, n_Б), i = 1,...,n && \text{(г)} \end{aligned} \qquad (16)$$

от Б - к А

$$\begin{aligned} n_A &= p_A q_A, && \text{(а)} \\ \varphi_A &= (p_A - 1)(q_A - 1), && \text{(б)} \\ d_A &= f_8(0, \varphi_A, \varphi_A), && \text{(в)} \\ c_{iБ} &= f_7(m_{iБ}, d_A, n_A), i = 1,...,n && \text{(г)} \end{aligned} \qquad (17)$$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$c_i = f_9(m_i, d, n), i = 1,...,n \qquad (18)$$

4.10. Алгоритм дешифрования

Алгоритм дешифрования схож с алгоритмом шифрования, описанном в разделе 4.9. Принятое зашифрованное сообщение необходимо возвести в степень e (закрытый ключ абонента, которому доставлено сообщение) по модулю n. Закрытый ключ является секретным и не передается в открытый канал связи. Данный ключ формируется с помощью чисел $p, q \in P$. Данная пара чисел является секретной для каждого абонента. Не секретным является число $n = pq$, которое легко вычислить, зная p и q. Однако разложить число n на составляющие является сложной задачей, которая требует много времени.

Входные параметры: $\vec{c} = \{c_1, c_2, ..., c_n\}$ - зашифрованное сообщение,

$e \in N$ - закрытый ключ абонента,

$n \in N$ - открытая часть ключа

Выходные параметры: $\vec{m} = \{m_1, m_2, ..., m_n\}$ - расшифрованное сообщение

Алгоритм дешифрования описывается уравнениями:

от А - к Б

$$\begin{aligned} n_Б &= p_Б q_Б, && \text{(а)} \\ \varphi_Б &= (p_Б - 1)(q_Б - 1), && \text{(б)} \\ d_Б &= f_8(0, \varphi_Б, \varphi_Б), && \text{(в)} \\ e_Б &= f_5(d_Б, \varphi_Б), && \text{(г)} \\ m_{iA} &= f_7(c_{iA}, e_Б, n_Б),\ i = 1,...,n && \text{(д)} \end{aligned} \qquad (19)$$

от Б - к А

$$\begin{aligned} n_A &= p_A q_A, && \text{(а)} \\ \varphi_A &= (p_A - 1)(q_A - 1), && \text{(б)} \\ d_A &= f_8(0, \varphi_A, \varphi_A), && \text{(в)} \\ e_A &= f_5(d_A, \varphi_A), && \text{(г)} \\ m_{iБ} &= f_7(c_{iБ}, e_A, n_A),\ i = 1,...,n && \text{(д)} \end{aligned} \qquad (20)$$

В дальнейшем данный алгоритм будем использовать в виде функции:

$$m_i = f_{10}(c_i, e, n),\ i = 1,...,n \qquad (21)$$

4.11. Алгоритм декодирования

Алгоритм декодирования является обратным по отношению к алгоритму кодирования, описанному в разделе 4.1. Данный алгоритм ставит в соответствие каждому цифровому символу алфавита Z_{44} символ алфавита A_{44} согласно таблице 3.

Входные параметры: $\vec{m} = \{m_1, m_2, ..., m_n\},\ n \in N$ - закодированное сообщение,

Выходные параметры: $\vec{a} = \{a_1, a_2, ..., a_n\},\ n \in N$ - декодированное сообщение;

В дальнейшем данный алгоритм будем использовать в виде функции:

$$a_i = f_{11}(m_i),\ i = 1, ..., n \qquad (22)$$

5. Разработка структурной схемы секретной связи с открытым ключом

В данном разделе разрабатывается структурная схема секретной связи с открытым ключом на основе алгоритма RSA. Структурная схема приведена на рисунке 1.

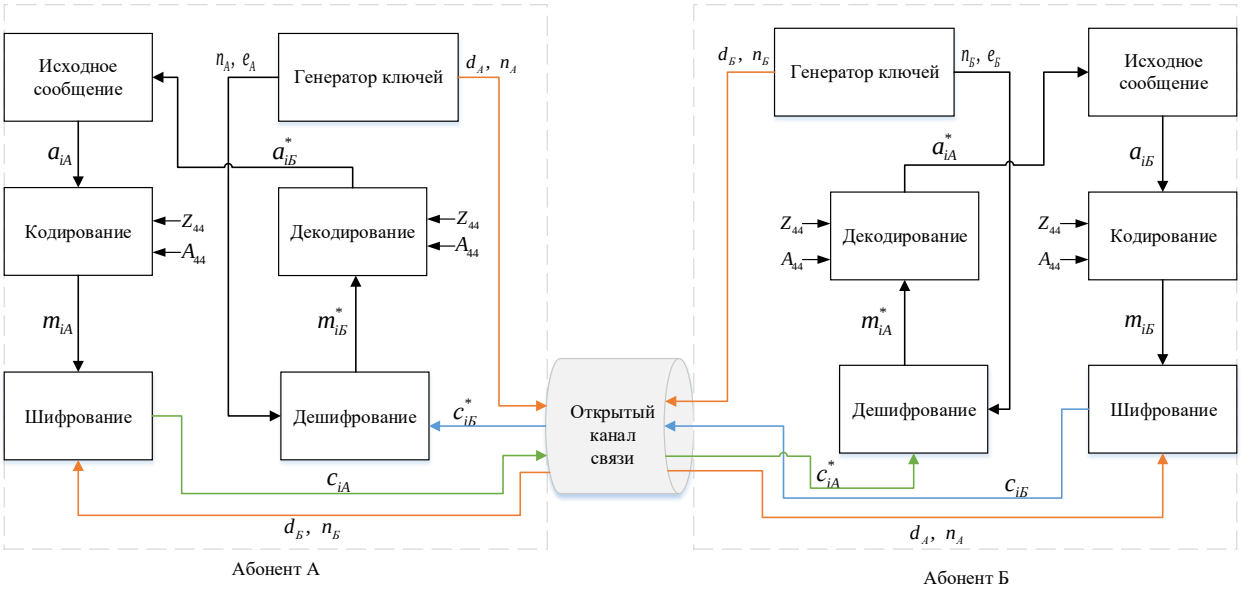

Рисунок 1. Структурная схема секретной связи с открытым ключом

Рассмотрим работу данной схемы на примере передачи сообщения от абонента А к абоненту Б. Абоненты генерируют ключи d, e, n. Ключи d и n необходимы для шифрования сообщений, они не являются секретными и передаются в открытый канал связи. Исходное сообщение абонента А $\vec{a}_A = \{a_1, a_2, ..., a_n\}$ кодируется в соответствии с формулой (1). После кодирования символы закодированного сообщения m_{iA} передаются в блок Шифрование, где в соответствием с формулой (16. г) шифруются, с помощью открытой пары ключей $d_Б, n_Б$ абонента Б, сгенерированными по формулам (16.а) и (16.в). После, символы зашифрованного сообщения c_{iA} передается абоненту Б по открытому каналу связи. Абонент Б, получив зашифрованное сообщение абонента А, дешифрует его в соответствии с формулой (19.д) с помощью своего закрытого ключа $e_Б$, сгенерированного с помощью формулы (19.г). После дешифрации сообщение m^*_{iA} поступает на декодер абонента Б, где согласно формуле (22) сообщение переводится из цифрового кода в буквенный.

Передача сообщения от абонента Б к абоненту А происходит аналогичным образом.

6. Разработка программных модулей

6.1. Функция кодирования

```
function [m] = my_A44_to_Z44(a)
% Функция кодирования исходного сообщения с использованием алфавита Z44
% Алгоритм работы данной функции изложен в разделе 4.1
% Функция работает в соответствии с формулой (1) раздела 4.1.
% ---------------------------------------------------------------------
% Входные параметры:
%  a - исходное сообщение
% ---------------------------------------------------------------------
% Выходные параметры:
%  m - закодированное сообщение
% ---------------------------------------------------------------------
   m = zeros(1, length(a));
   alphabet = 'АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ!?.,:;()""-';
   for i = 1:length(a)
      if a(i) == ' '
         m(i) = 0;
      else
         m(i) = find(alphabet==a(i));
      end
   end
end
```

6.2. Функция, реализующая классический алгоритм Евклида

```
function [r] = my_classic_Euclid(a, b)
% Функция вычисления НОД с помощью классического алгоритма Евклида
% Алгоритм работы данной функции представлен в разделе 4.2
% Функция работает в соответствии с формулой (3) раздела 4.2
% ---------------------------------------------------------------------
% Входные параметры:
%  a - первое число
%  b - второе число
% ---------------------------------------------------------------------
% Выходные параметры:
%  r - НОД чисел a и b
% ---------------------------------------------------------------------
   while mod(a,b) ~= 0
      c = b;
      b = mod(a,b);
      a = c;
   end
   r = b;
end
```

6.3. Функция нахождения взаимно простого числа

```
function [r] = my_mut_simple(a)
% Функция определения взаимно простого числа с a
% Алгоритм работы данной фукции представлен в разделе 4.3
% Функция работает в соответствии с формулой (5) раздела 4.3
% ---------------------------------------------------------------------
% Входные параметры:
%  a - натуральное число
% ---------------------------------------------------------------------
% Выходные параметры:
%  e – взаимно простое число с a
% ---------------------------------------------------------------------

   for r = 2:a
      if my_classic_Euclid(a,r) == 1
         break
      end
   end
end
```

6.4. Функция, реализующая обобщенный алгоритм Евклида

```
function [c, x, y] = my_ob_Euclid(a,b)
% Функция, реализующая обобщенный алгоритм Евклида
% Функция работает в соответствии с формулой (7) раздела 4.4
% ---------------------------------------------------------------------
% Входные параметры:
%  a - первое число
%  b - второе число
%  a, b - натуральные числа
% ---------------------------------------------------------------------
% Выходные параметры:
%  c - НОД чисел a и b - натуральное число
%  x - коэффициент при a
%  y - коэффициент при b
%  x, y - целые числа
% ---------------------------------------------------------------------

   u = [a 1 0];
   v = [b 0 1];
   while v(1) ~= 0
      q = floor(u(1)/v(1));
      t = [mod(u(1),v(1)) u(2)-q*v(2) u(3)-q*v(3)];
      u = v;
      v = t;
   end
   c = u(1);
   x = u(2);
   y = u(3);
end
```

6.5. Функция вычисления обратного по модулю числа

```
function [d] = my_inv_mod(c, n)
% Функция, вычисляющая обратное число по модулю
% Алгоритм работы функции представлен в разделе 4.5
% Функция работает в соответствии с формулой (9) раздела 4.5
% ------------------------------------------------------------------------
% Входные параметры:
%  c - число
%  n - модуль
%  c, n - натуральные числа
% ------------------------------------------------------------------------
% Выходные параметры:
%  d - инверсия числа c по модулю n - натуральное число
% ------------------------------------------------------------------------
   [~, ~, y] = my_ob_Euclid(n,c);
   d = mod(y, n);
end
```

6.6. Функция перевода десятичного числа в двоичное

```
function [r] = my_dec_to_bin(num)
% Функция перевода десятичного числа в двоичное
% Алгоритм работы функции представлен в разделе 4.6
% Функция работает в соответствии с формулой (11) раздела 4.6
% ------------------------------------------------------------------------
% Входные параметры:
%  a - целое число в десятичном виде
% ------------------------------------------------------------------------
% Выходные параметры:
%  x - представление числа a в двоичном виде
% ------------------------------------------------------------------------
   n = floor(log(num)/log(2)) + 1;
   r = zeros(1,n);
   for i = 1:n
      r(i) = mod(num, 2);
      num = fix(num/2);
  end
end
```

6.7. Функция эффективного возведения в степень числа

```
function [y] = my_stepen(a, x, p)
% Данная функция описывает алгоритм эффективного возведения в степень
% справа-налево
% Алгоритм работы данной функции представлен в разделе 4.7
% Функция работает в соответствии с формулой (13) раздела 4.7
%--------------------------------------------------------------------------
% Входные параметры:
%  a, x, p - целые числа
% Выходные переменные:
%  y = a^x (mod p)
%  y - целое число
%--------------------------------------------------------------------------
   y = 1; s = a;
   x_tmp = my_dec_to_bin(x);
   for i = 1:length(x_tmp)
      if x_tmp(i) == 1
         y = mod(y*s, p);
      end
      s = mod(s*s, p);
   end
end
```

6.8. Функция генерации ключей

```
function [n, phi, e, d] = my_key_gen(p, q)
% Функция вычисления открытого ключа для алгоритма RSA
% Алгоритм работы функции представлен в разделе 4.8
% Функция работает в соответствии с формулой (15) раздела 4.8
%--------------------------------------------------------------------------
% Входные параметры:
%  p, q - простые числа
%--------------------------------------------------------------------------
% Выходные параметры:
%  n - произведение p и q
%  phi - функция Эйлера от p и q
%  e - закрытая экспонента
%  d - открытая экспонента
%--------------------------------------------------------------------------
   n = p*q;
   phi = (p-1)*(q-1);
   k_min = 0; k_max = phi;
   d0 = k_min;
   d = k_min + fix((k_max - k_min) * rand);
   while d == d0 || my_classic_Euclid(d, phi) ~= 1
      d = k_min + fix((k_max - k_min) * rand);
   end
   e = my_inv_mod(d, phi);
end
```

6.9. Функция шифрования

```
function [c] = my_shifr_rsa(m, d, n)
% Функция шифрования сообщения по алгоритму RSA
% Алгоритм данной функции представлен в разделе 4.9
% Данная функция работает на основании формулы (18) раздела 4.9
%--------------------------------------------------------------------------
% Входные параметры:
%  m - исходное закодированное сообщение
%  d, n - открытые части ключей
%--------------------------------------------------------------------------
% Выходные параметры:
%  c - зашифрованное сообщение
%--------------------------------------------------------------------------
   c = zeros(1, length(m));
   for i=1:length(m)
      c(i) = my_stepen(m(i), d, n);
   end
end
```

6.10. Функция дешифрования

```
function [m] = my_deshifr_rsa(c, e, n)
% Функция дешифрования сообщения по алгоритму RSA
% Алгоритм данной функции представлен в разделе 4.10
% Данная функция работает на основании формулы (21) раздела 4.10
%--------------------------------------------------------------------------
% Входные параметры:
%  c - зашифрованное сообщение
%  n - открытая часть ключа
%  e - закрытая часть ключа
%--------------------------------------------------------------------------
% Выходные параметры:
%  m - исходное закодированное сообщение
%--------------------------------------------------------------------------
   m = zeros(1, length(c));
   for i =1:length(c)
      m(i) = my_stepen(c(i), e, n);
   end
end
```

6.11. Функция декодирования

```
function [a] = my_Z44_to_A44(m)
% Функция декодирования исходного сообщения с использованием алфавита Z44
% Алгоритм работы функции приведен в разделе 4.11
% Функция работает в соответствии с формулой (22) раздела 4.11
% ------------------------------------------------------------------------
% Входные параметры:
%  m - закодированное сообщение
% ------------------------------------------------------------------------
% Выходные параметры:
%  a - декодированное сообщение
% ------------------------------------------------------------------------
   a = blanks(length(m));
   alphabet = 'АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ!?.,:;()""-';
   for i = 1:length(m)
      if m(i) == 0
         a(i) = ' ';
      else
         a(i) = alphabet(m(i));
      end
   end
end
```

7. Разработка тестовой программы

7.1. Код программы

```
% Исходные данные
p = 193; q = 347;

disp('-----------------------------------------------------------------')
a = 'ЗАЙЦЕВ ДАНИИЛ РОМАНОВИЧ'; % Исходное сообщение
disp(['Исходное сообщение: ', a]);
disp('-----------------------------------------------------------------')
disp('Исходные данные:');
disp(['p = ', num2str(p)]);
disp(['q = ', num2str(q)]);
disp('-----------------------------------------------------------------')

% Генерация ключей
disp('Ключи:');
[n, phi, e, d] = my_key_gen(p, q);
disp(['n = pq = ', num2str(n)]);
disp(['phi = (p - 1)(q - 1) = ', num2str(phi)]);
disp(['Закрытый ключ e = ', num2str(e)]);
disp(['Открытый ключ d = ', num2str(d)]);
disp('-----------------------------------------------------------------')

% Кодирование сообщения
m = my_A44_to_Z44(a);
disp('Закодированное сообщение: ');
disp(m)
disp('-----------------------------------------------------------------')

% Шифрование сообщения
c = my_shifr_rsa(m, d, n);
disp('Зашифрованное сообщение :')
disp(c);
disp('-----------------------------------------------------------------')

% Дешифрования сообщения
m_ = my_deshifr_rsa(c, e, n);
disp('Дешифрованное сообщение:');
disp(m_);
disp('-----------------------------------------------------------------')

% Декодирование сообщения
a_ = my_Z44_to_A44(m_);
disp('Декодированное сообщение: ')
disp(a_);
disp('-----------------------------------------------------------------')
```

7.2. Результат работы программы

```
>> my_test
```

Исходное сообщение: ЗАЙЦЕВ ДАНИИЛ РОМАНОВИЧ

Исходные данные:

p = 193

q = 347

Ключи:

n = pq = 66971

phi = (p - 1)(q - 1) = 66432

Закрытый ключ e = 60257

Открытый ключ d = 10145

Закодированное сообщение:

8 1 10 23 6 3 0 5 1 14 9 9 12 0 17 15 13 1 14 15 3 9 24

Зашифрованное сообщение :

Columns 1 through 14

62733 1 43889 35342 24436 44393 0 26600 1 62739 49803 49803 30439 0

Columns 15 through 23

5134 21128 57501 1 62739 21128 44393 49803 50976

Дешифрованное сообщение:

8 1 10 23 6 3 0 5 1 14 9 9 12 0 17 15 13 1 14 15 3 9 24

Декодированное сообщение:

ЗАЙЦЕВ ДАНИИЛ РОМАНОВИЧ

---

8. Вывод

В данной курсовой работе была разработана система секретной связи с открытым ключом на основе алгоритма RSA. В ходе работы были разработаны и запрограммированы следующие алгоритмы:

1) Алгоритм кодирования
2) Классический алгоритм Евклида
3) Алгоритм нахождения взаимно простого числа
4) Обобщенный алгоритм Евклида
5) Алгоритм вычисления обратного по модулю числа
6) Алгоритм перевода десятичного числа в двоичное
7) Алгоритм эффективного возведения в степень
8) Алгоритм генерации ключей
9) Алгоритм шифрования
10) Алгоритм дешифрования
11) Алгоритм декодирования

Каждый алгоритм реализован в виде отдельного программного модуля на языке Matlab. Для тестирования системы секретной связи было закодировано, а в последствии зашифровано сообщение «ЗАЙЦЕВ ДАНИИЛ РОМАНОВИЧ», а после успешно дешифровано и декодировано. Получив после декодирования расшифрованного сообщения исходное сообщение можно утверждать, что система работает корректно.

Основным достоинством алгоритма RSA является то, что злоумышленнику, перехватившему зашифрованное сообщение, но не знающему закрытый ключ, потребуется большие вычислительные мощности, а также довольно много времени для дешифрования перехваченного сообщения. При использовании больших чисел P и Q вычисление закрытого ключа становится почти невозможным.

Однако злоумышленник может подменить открытый ключ одного из абонентов, и выдавать себя за одного из абонентов, не являясь им. Данный недостаток является основным недостатком этого алгоритма.

В настоящий момент система алгоритм RSA используется в системах цифровой подписи и для защиты программного обеспечения. Так как скорость шифрования сообщений невелика, сообщения зачастую шифруют быстрыми симметричными алгоритмами, а с помощью RSA шифруют сеансовый ключ, таким образом реализуется гибридная криптосистема.

Список используемой литературы

1) Конспект лекций по Основам Криптографии
2) Задания и методические указания к выполнению КР по ОК 21 02 19
3) Санников В. Г. «Введение в теорию и методы криптографической защиты информации: учебное пособие» МТУСИ, 2009
4) Рябко Б. Я., Фионов А. Н. «Криптографические методы защиты информации» Горячая линия – Телеком, 2005
5) Потемкин В. Г., MATLAB: Справочное пособие. М.: Диалог-МИФИ., 1997
6) https://ru.wikipedia.org/wiki/RSA
7) https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом