Министерство образования и науки Российской федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**«**Воронежский государственный лесотехнический университет»

имени Г.Ф. Морозова

**ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И ЗАЩИТА ИНФОРМАЦИИ**

Методические указания к выполнению

лабораторных работ



Воронеж 2016

**УДК 004.056.5**

Чевычелов Ю.А.

Информационная безопасность и защита информации [Текст]: Методические указания к выполнения лабораторных работ для студентов специальности 09.03.02 - Информационные системы и технологии./ Ю.А. Чевычелов; М-во образования и науки РФ ФГОБУ ВО «ВГЛТУ» им. Г.Ф.Морозова. Воронеж. 2016

Указания содержат методический материал к выполнению лабораторных работ по дисциплине «Информационная безопасность т защита информации». Защиту информации в настоящее время необходимо рассматривать как часть интегрированной системы безопасности. Актуальность вопросов защиты информации обусловлена двумя обстоятельствами: первое – современные системы охранной безопасности, применяемые для противодействия проник- новениям на объекты собственности являются сложными информационно- телекоммуникационными системами, базирующимися на современных ин- формационных технологиях; второе – информация, хранящаяся и обрабаты- ваемая в информационных системах, также является объектом охраны

**Лабораторная работа 1**

**Тема: Простые числа.**

**Число́** — основное понятие [математики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), используемое для [количественной](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE) характеристики, сравнения, [нумерации](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F)) и их частей. Письменными знаками для обозначения чисел служат [цифры](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%B0), а также [символы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB) математических [операций](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Возникнув ещё в [первобытном обществе](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%BE%D0%B1%D1%8B%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE) из потребностей [счёта](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), понятие числа с развитием науки значительно расширилось.

**Из теории простых чисел**

Возникновение теории чисел принято связывать с именем великого греческого мыслителя **Пифагора**. Жил он в шестом веке до нашей эры и был современником Конфуция и Лао Цзы, Будды и Заратуштры, а также ряда библейских пророков. Будучи, главным образом, философом, Пифагор искал естественную связь между окружающей действительностью, воспринимаемой органами чувств, и человеческим разумом, познающим мир. Согласно Пифагору, таким связующим звеном и является *число*. Действительно, с одной стороны, число есть вполне абстрактное понятие, являющееся продуктом нашего мозга. Но, с другой стороны, всё, происходящее в мире, может быть в той или иной степени измерено, а значит, представлено в числах. Главная мысль Пифагора: в мире царит гармония, и выражена эта гармония – в числах. А отсюда следовало, что для того чтобы прикоснуться к этой гармонии, нужно изучать числа.

Конечно, с числами люди сталкивались в повседневной жизни однако смотрели на них исключительно как на средство решения тех или иных практических задач. Пифагор же провозгласил, что числа, будучи основой всего сущего, должны быть непосредственной целью исследования. Возможно, это и стало началом всей математики.

Первые шаги, связанные с исследованием любого класса объектов, обычно направлены на выявление присущих им специфических свойств. Стоило взглянуть на числа, как на непосредственный предмет исследования, как выяснилось, что они в чем-то различаются между собой. Сразу же бросается в глаза то естественное обстоятельство, что их можно сравнивать между собой: 5 больше, чем 2, но меньше, чем 10. Собственно, на этом и основано их практическое применение. Существенно более глубоким свойством оказывается возможность разложения чисел на более простые (меньшие по значению) сомножители.

Понятно, что любое число делится без остатка само на себя и на единицу. Однако подавляющее большинство чисел имеют и качественно иные представления, например, 8 = 2 × 4 или 15 = 3 × 5. Таким образом, все числа (речь, понятно, идет о натуральных числах: 1, 2, 3, …) можно разбить на два класса. *Простые числа* допускают только тривиальное представление, а *составные числа* разлагаются в произведение простых сомножителей. Упомянутые выше числа 8 и 15 естественно оказываются составными. Простыми числами являются 2, 3, 5, 7, 11, 13, 17 и т.д. Число 1 играет особую роль, к простым числам относить его не принято. Первым делом хотелось бы узнать, как много имеется простых чисел? Ответ на этот вопрос дается **Евклидом**, жившем в третьем веке до нашей эры в Александрии. В его знаменитых "*Началах*" приводятся следующие рассуждения. Предположим, что существует лишь конечное множество простых чисел, исчерпывающиеся значениями *р*1 , *р*2 ,…, *рk* . Тогда число *p* = *p*1 × *p*2 ×...× *pk* +1 никак не может делиться без остатка на выше указанные числа, поскольку непременно имеет единицу в качестве такового остатка. Отсюда следует, что число *р* само является простым, а значит, предположение Евклида 4 в. до н.э. о том, что все простые числа перечислены в указанном выше перечне, не соответствует действительности. Тем самым ***множество простых чисел оказывается бесконечным***. С Евклидом связана также и *основная теорема арифметики*, согласно которой любое натуральное число (кроме 1) однозначно разлагается в произведение простых чисел, например, 12 = 2 × 2 × 3, а 35 = 5 × 7.т.е. каждое [натуральное число](http://dic.academic.ru/dic.nsf/ruwiki/5311), большее единицы, представимо в виде произведения простых чисел, причём единственным способом с точностью до порядка следования сомножителей. Таким образом, простые числа — элементарные «строительные блоки» натуральных чисел.

[**Натуральные числа**](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE), получаемые при естественном счёте; [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) натуральных чисел обозначается\mathbb{N}. То есть \mathbb{N}=\left\{1, 2, 3, ...\right\} (иногда к множеству натуральных чисел также относят [ноль](https://ru.wikipedia.org/wiki/0_(%D1%87%D0%B8%D1%81%D0%BB%D0%BE)), то есть \mathbb{N}=\left\{0, 1, 2, 3, ...\right\}. Натуральные числа [замкнуты](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0)) относительно [сложения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [умножения](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) (но не [вычитания](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B5) или [деления](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))). Сложение и умножение натуральных чисел [коммутативны](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и [ассоциативны](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), а умножение натуральных чисел [дистрибутивно](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) относительно сложения и вычитания.

[**Целые числа**](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE), получаемые объединением натуральных чисел с множеством чисел [противоположных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%B8%D0%B2%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) натуральным и нулём, обозначаются \mathbb{Z}=\left\{...-2, -1, 0, 1, 2, ...\right\}. Любое целое число можно представить как разность двух натуральных. Целые числа замкнуты относительно сложения, вычитания и умножения (но не деления). Такая алгебраическая структура называется [кольцом](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%8C%D1%86%D0%BE_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))

Представление натурального числа в виде произведения простых называется разложением на простые или [факторизацией числа](http://dic.academic.ru/dic.nsf/ruwiki/1609796). На настоящий момент неизвестны [полиномиальные алгоритмы](http://dic.academic.ru/dic.nsf/ruwiki/1099932) факторизации чисел, хотя и не доказано, что таких алгоритмов не существует. На предполагаемой большой [вычислительной сложности](http://dic.academic.ru/dic.nsf/ruwiki/855899) задачи факторизации базируется криптосистема [RSA](http://dic.academic.ru/dic.nsf/ruwiki/20735) и некоторые другие.

**Алгоритмы нахождения простых чисел.**

Задача 1

Составить программу, которая будет проверять, является ли введенное число простым. Самый простой путь решения этой задачи - проверить, имеет ли данное число n (n > 2) делители в интервале [2; n-1]. Если делители есть, число n - составное, если - нет, то - простое. При реализации алгоритма разумно делать проверку на четность введенного числа, поскольку все четные числа делятся на 2 и являются составными числами, то, очевидно, что нет необходимости искать делители для этих чисел. Ввести логическую переменную flag, в программе выступаете в роли "флаговой" переменной и повышает наглядность программы, так, если flag = true, то n -простое число; если у числа n есть делители, то "флаг выключается" с помощью оператора присваивания flag:=false, таким образом, если flag = false, то n - составное число.

Пример реализации на VB6.

For j = 3 To N - 1 Step 2

k = m Mod j

If k = 0 And j <> m Then

flag = False

Form1.BackColor = RGB(256, 0, 0)

Text1.FontSize = 16

Text1.Text = "Число составное"

Exit For

ElseIf k = 0 And j = m Then

flag = True

Form1.BackColor = RGB(0, 256, 0)

Text1.FontSize = 16

Text1.Text = "Введено простое число"

Exit For

End If

m = Val(InputBox("Введите число ",”Ввод данных”, 11))

Next j

MsgBox " k= " & Str(r) & " " & "j = " & Str(j)

MsgBox "Работа программы окончена"

Задача 2. Нахождение простых чисел в заданном интервале.

Составить программу, которая напечатает все простые числа в заданном интервале [2, m], для m>3 и подсчитает их количество. Для реализации данного алгоритма необходимо проверить каждое число, находящееся в данном интервале, - простое оно или нет. Однако для этого машине пришлось бы потратить много времени. подумать, каким образом можно оптимизировать алгоритм, описанный в задаче 1. При реализации решения выполнять следующие действия:

1. рассматривать только нечетные числа;
2. Счетчик чисел будет находиться в переменной k. Когда очередное простое число найдено, увеличивать k на 1, Простые числа выводятся по 10 в строке, как только значение счетчика становится кратным 10, курсор переводится на новую строку.

Реализовать алгоритм задачи 2 с выводом простых чисел в выходной файл по 10 в строке.

For j = m To N - 1 Step 2

For i = 3 To j Step 2

k = j Mod i

If k = 0 And j <> i Then

flag = False

Exit For

ElseIf k = 0 And j = i Then

flag = True

l = l + 1

If l <= 10 Then

Print j;

ElseIf l > 10 Then

l = 0

Print

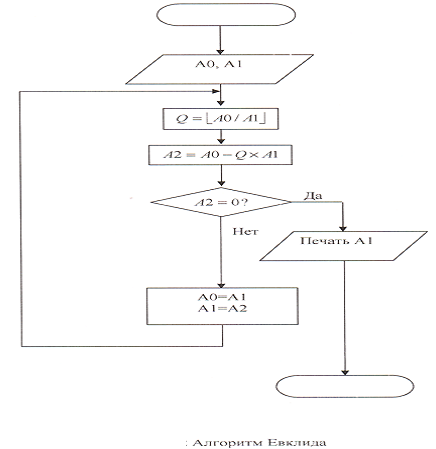
End If

End If

Next i

Next

MsgBox "Работа программы завершена"

**

*НОД(a,n)=1*

Одним из способов вычислить наибольший общий делитель двух чисел является **алгоритм Эвклида.** Эвклид описал этот алгоритм в своей книге, *Элементы,* написанной в 300 году до нашей эры. Он не изобрел его. Историки считают, что этот алгоритм лет на 200 старше. Это самый древний нетривиальный алгоритм, который дошел до наших дней, и он все еще хорош. Кнут описал алгоритм и его современные модификации y на языке С:

/\* возвращает НОД(gcd) x и у \*/ int gcd (int x, int у)

{

int g;

if (x < 0)

x = -x;

if (y < 0)

у = -у;

if (x + у == 0

ERROR;

g=y;

While (x>0){

q=x;

x=y % x;

y=q;

}

return q;

}

Задача 3 Используя алгоритм Евклида найти простые числа при N = 150,...

For a = 3 To 150 Step 2

x = a

If a Mod 3 = 0 Or a Mod 5 = 0 Or a Mod 7 = 0 Or a Mod 11 = 0 Then GoTo 20

For b = 3 To 97 Step 2

c = EVKLYD(x, b)

If c = 1 Then

k = k + 1

Print a;

If k = 10 Then Print k = 0

End If

Exit For

End If

Next b

20 Next a

В отчете по лабораторной работе должно содержаться:

1. Титульный лист в соответствии со стандартом оформления титульного листа работы..
2. Задание на лабораторную работу.
3. Краткую теорию.
4. Ход выполнения работы: алгоритмы, программное обеспечение (исходный код
5. Результаты тестирования.
6. Вид формы приложения.
7. Выводы.

**Лабораторная работа 2**

1. ВЗАИМНО ПРОСТЫЕ ЧИСЛА

**Взаимно простые числа – определение и примеры**

Понятие взаимно простых чисел дается как для двух целых чисел, так и для их большего числа. Сначала приведем **определение двух взаимно простых чисел**. Это определение дается через [наибольший общий делитель чисел](http://www.cleverstudents.ru/divisibility/nod.html).

*Определение.*

Два целых числа *a* и *b* называются **взаимно простыми**, если их наибольший общий делитель равен единице, то есть, *НОД(a, b)=1*.

Из определения взаимно простых чисел следует, что два взаимно простых числа имеют лишь один положительный общий делитель, который равен единице. А всего общих делителей у двух взаимно простых чисел два – это числа *1* и *−1*.

**Примеры взаимно простых чисел**.

Числа *5* и *11* являются взаимно простыми. Действительно, и *5* и *11* – [простые числа](http://www.cleverstudents.ru/divisibility/prime_and_composite_numbers.html), следовательно, их положительным общим делителем является только число *1*, что подтверждает взаимную простоту чисел *5* и *11*.

Заметим, что два простых числа всегда являются взаимно простыми. Однако, два числа не обязательно должны быть простыми, чтобы быть взаимно простыми. Либо одно из них, либо они оба могут быть составными и при этом являться взаимно простыми. Приведем пример, иллюстрирующий это высказывание.

Два составных числа *8* и *−9* являются взаимно простыми. Обоснуем это. Для этого найдем наибольший общий делитель этих чисел, записав все делители чисел *8* и *−9.* Делителями восьмерки является любое из чисел *1*, *±2*, *±4*, *±8*; все делители *−9* есть числа*±1±3*,*±9*. Следовательно, *НОД(8, −9)=1*, поэтому, по определению *8* и*−9*– два взаимно простых числа.

А вот числа *45* и *500* не являются взаимно простыми, так как имеют положительный общий делитель, отличный от единицы, которым является число *5* (делимость чисел*45* и *500* на *5* очевидна, если знать [признак делимости на 5](http://www.cleverstudents.ru/divisibility/divisibility_rule_for_5.html)). Другой парой не взаимно простых чисел является пара *3* и *−201*, так как *3* есть их общий положительный делитель (делимость числа *−201* на *3* легко устанавливается при помощи [признака делимости на 3](http://www.cleverstudents.ru/divisibility/divisibility_rule_for_3.html)).

Часто встречаются задания, в которых требуется доказать, что данные целые числа являются взаимно простыми. Доказательство сводится к вычислению наибольшего общего делителя данных чисел и проверке НОД на его равенство единице. Полезно также перед вычислением НОД заглянуть в [таблицу простых чисел](http://www.cleverstudents.ru/divisibility/prime_and_composite_numbers.html#table_of_primes): вдруг исходные целые числа являются простыми, а мы знаем, что наибольший общий делитель простых чисел равен единице. Рассмотрим решение примера.

*Пример.*

Докажите, что числа *84* и *275* являются взаимно простыми.

*Решение.*

Очевидно, что данные числа не являются простыми, поэтому мы не можем сразу говорить о взаимной простоте чисел *84* и *275*, и нам придется вычислять НОД. Используем [алгоритм Евклида для нахождения НОД](http://www.cleverstudents.ru/divisibility/nod_finding.html#Euclids_algorithm): *275=84·3+23*,*84=23·3+15*, *23=15·1+8*, *15=8·1+7*,  *8=7·1+1*,*7=7·1*, следовательно, *НОД(84, 275)=1*. Этим доказано, что числа *84* и *275* взаимно простые.

Определение взаимно простых чисел можно расширить для трех и большего количества чисел.

*Определение.*

Целые числа *a1, a2, …, ak*, *k>2* называются **взаимно простыми**, если наибольший общий делитель этих чисел равен единице.

Из озвученного определения следует, что если некоторый набор целых чисел имеет положительный общий делитель, отличный от единицы, то данные целые числа не являются взаимно простыми.

Приведем примеры. Три целых числа *−99*, *17* и *−27* являются взаимно простыми. Любая совокупность простых чисел составляет набор взаимно простых чисел, к примеру, *23 11*, *19 151*, *293* и *677*– взаимно простые числа. А четыре числа 1*2*,*−9*, *900* и*−72* не являются взаимно простыми, так как они имеют положительный общий делитель *3*, отличный от*1*. Числа *1 85* и *187*  тоже не взаимно простые, так как каждое из них делится на *17*.

Обычно далеко не очевидно, что некоторые числа являются взаимно простыми, и этот факт приходится доказывать. Для выяснения, являются ли данные числа взаимно простыми, приходится находить наибольший общий делитель этих чисел, и на основании определения взаимно простых чисел делать вывод.

*Пример.*

Являются ли числа *331*, *463* и *733* взаимно простыми?

*Решение.*

Заглянув в таблицу простых чисел, мы обнаружим, что каждое из чисел *331*,*463* и *733* – простое. Следовательно, они имеют единственный положительный общий делитель – единицу. Таким образом, три числа *331*,*463* и *733* есть взаимно простые числа.

*Ответ:*

Да.

*Пример.*

Докажите, что числа *−14*, *105*,*−2 107* и *−91* не являются взаимно простыми.

*Решение.*

Чтобы доказать, что данные числа не взаимно простые, можно найти их НОД и убедиться, что он не равен единице. Так и поступим.

Так как делители целых отрицательных чисел совпадают с делителями соответствующих [противоположных чисел](http://www.cleverstudents.ru/numbers/opposite_numbers.html), то *НОД(−14, 105, 2 107, −91)=НОД(14, 105, 2 107, 91)*. Обратившись к материалу статьи [нахождение наибольшего общего делителя трех и большего количества чисел](http://www.cleverstudents.ru/divisibility/nod_finding.html#3_or_more), выясняем, что *НОД(14, 105, 2 107, 91)=7*. Следовательно, наибольший общий делитель исходных чисел равен семи, поэтому эти числа не являются взаимно простыми.

**Свойства взаимно простых чисел**

Взаимно простые числа обладают рядом свойств. Рассмотрим основные **свойства взаимно простых чисел**.

1. Числа, полученные при делении целых чисел *a* и *b* на их наибольший общий делитель, являются взаимно простыми, то есть, *a:НОД(a, b* и *b:НОД(a, b)*– взаимно простые.

Это свойство мы доказали, когда разбирали [свойства НОД](http://www.cleverstudents.ru/divisibility/nod.html#properties).

Рассмотренное свойство взаимно простых чисел позволяет находить пары взаимно простых чисел. Для этого достаточно взять два любых целых числа и разделить их на наибольший общий делитель, полученные числа будут взаимно простыми.

1. Для того чтобы целые числа *a* и *b* были взаимно простыми необходимо и достаточно, чтобы существовали такие целые числа *u0* и *v0*, что *a·u0+b·v0=1*.

Докажем сначала необходимость.

Пусть числа *a* и *b* взаимно простые. Тогда по определению взаимно простых чисел *НОД(a, b)=1*. А из свойств НОД мы знаем, что для целых чисел *a* и *b* верно соотношение Безу *a·u0+b·v0=НОД(a, b)*. Следовательно, *a·u0+b·v0=1*.

Осталось доказать достаточность.

Пусть верно равенство *a·u0+b·v0=1*. Так как *НОД(a, b)* делит и *a* и *b*, то*НОД(a, b)* в силу [свойств делимости](http://www.cleverstudents.ru/divisibility/divisibility.html#properties) должен делить сумму *a·u0+b·v0*, а значит, и единицу. А это возможно только когда *НОД(a, b)=1*. Следовательно,*a* и *b* – взаимно простые числа.

1. Следующее свойство взаимно простых чисел таково: если числа *a* и *b* взаимно простые, и произведение *ac* делится на *b*, то *c* делится на *b*.

Действительно, так как *a* и *b* взаимно простые, то из предыдущего свойства мы имеем равенство *a·u0+b·v0=1*. Умножив обе части этого равенства на *c*, имеем *ac\*u0+bc\*v0=c*. Первое слагаемое суммы *ac\*u0+bc\*v0* делится на *b*, так как *ac* делится на *b* по условию, второе слагаемое этой суммы также делится на *b*, так как один из множителей равен *b*, следовательно, вся сумма делится на *b*. А так как сумма *a·c·u0+b·c·v0* равна *c*, то и *c* делится на *b*.

1. Если числа *a* и *b* взаимно простые, то *НОД(a·c, b)=НОД(c, b)*.

Покажем, во-первых, что *НОД(a·c, b)* делит *НОД(c, b)*, а во-вторых, что*НОД(c, b)* делит *НОД(a·c, b)*, это и будет доказывать равенство*НОД(a·c, b)=НОД(c, b)*.

*НОД(a·c, b)* делит и *a·c* и *b*, а так как *НОД(a·c, b)* делит *b*, то он также делит и *b·c*. То есть, *НОД(a·c, b)* делит и *a·c* и *b·c*, следовательно, в силу свойств наибольшего общего делителя он делит и *НОД(a·c, b·c)*, который по свойствам НОД равен *c·НОД(a, b)=c*. Таким образом, *НОД(a·c, b)* делит и *b* и*c*, следовательно, делит и *НОД(c, b)*.

С другой стороны, *НОД(c, b)* делит и *c* и *b*, а так как он делит *с*, то также делит и *a·c*. Таким образом, *НОД(c, b)* делит и *a·c* и *b*, следовательно, делит и *НОД(a·c, b)*.

Так мы показали, что *НОД(a·c, b)* и *НОД(c, b)* взаимно делят друг друга, значит, они равны.

1. Если каждое из чисел *a1, a2, …, ak* взаимно просто с каждым из чисел*b1, b2, …, bm* (где *k* и *m* – некоторые натуральные числа), то произведения*a1·a2·…·ak* и *b1·b2·…·bm* есть взаимно простые числа, в частности, если*a1=a2=…=ak=a* и *b1=b2=…=bm=b*, то *ak* и *bm* – взаимно простые числа.

Предыдущее свойство взаимно простых чисел позволяет нам записать ряд равенств вида *НОД(a1·a2·…·ak, bm)=НОД(a2·…·ak, bm)=…=НОД(ak, bm)=1*, где последний переход возможен, так как *ak* и *bm* взаимно простые числа по условию. Итак, *НОД(a1·a2·…·ak, bm)=1*.

Теперь, обозначив *a1·a2·…·ak=A*, имеем *НОД(b1·b2·…·bm, a1·a2·…·ak)=НОД(b1·b2·…·bm, A)==НОД(b2·…·bm, A)=… =НОД(bm, A)=1*  Так мы получили равенство *НОД(b1·b2·…·bm, a1·a2·…·ak)=1*, которое доказывает, что произведения*a1·a2·…·ak* и *b1·b2·…·bm* являются взаимно простыми числами.

**Попарно простые числа – определения и примеры**

Через взаимно простые числа дается **определение попарно простых чисел**.

*Определение.*

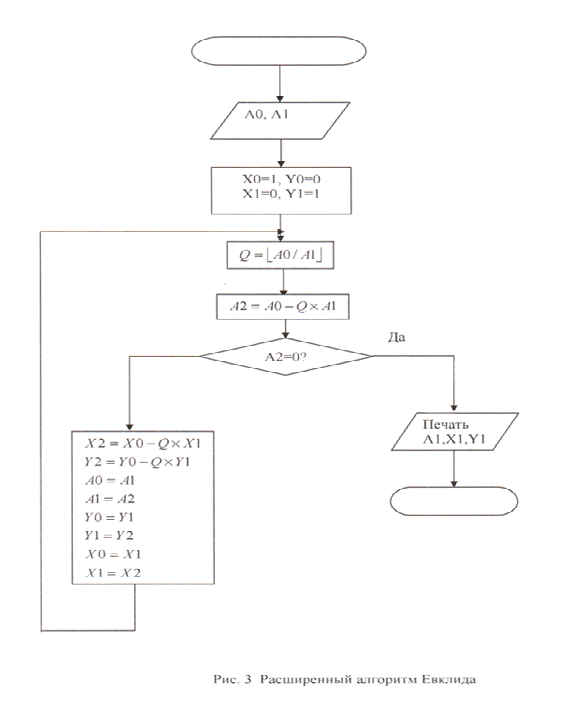
Целые числа *a1, a2, …, ak*, каждое из которых взаимно просто со всеми остальными, называют **попарно простыми числами**.

Приведем пример попарно простых чисел. Числа *14*, *9*, *17*, и *−25* – попарно простые, так как пары чисел *14* и *9*, *14* и *17*, *14* и *−25*, *9* и *17*, *9* и *−25*, *17* и *−25* представляют собой взаимно простые числа. Здесь же заметим, что попарно простые числа всегда являются взаимно простыми.

С другой стороны, взаимно простые числа далеко не всегда являются попарно простыми, это подтверждает следующий пример. Числа *8*, *16*, *5* и *15* не являются попарно простыми, так как числа *8* и *16* не взаимно простые. Однако, числа *8*, *16*, *5* и *15* – взаимно простые. Таким образом, *8*, *16*, *5* и *15* – взаимно простые числа, но не попарно простые.

Следует особо выделить совокупность некоторого количества простых чисел. Эти числа всегда являются и взаимно простыми и попарно простыми. Например, *71*, *443*, *857*, *991* – и попарно простые, и взаимно простые числа.

Также понятно, что когда речь идет о двух целых числах, то для них понятия «попарно простые» и «взаимно простые» совпадают.



Задание на выполнение работы

1.Реализовать алгоритм формирования массива взаимно простых чисел.

Тестирование программы выполнить на десяти наборах данных.

Программное обеспечение вывести в файл.

2. Создать программу реализующую решение уравнения m\*х + d\*e = 1 на базе модифицированного алгоритма Евклида.

Должна быть обеспечена наглядность выполнения алгоритма. Для создания программного обеспечения и подтверждения его работоспособности тестирование проводить не менее чем на 10 различных наборах данных.

В отчете по лабораторной работе должно содержаться:

1. Титульный лист в соответствии со стандартом оформления титульного листа работы..
2. Задание на лабораторную работу.
3. Краткую теорию.
4. Ход выполнения работы: алгоритмы, программное обеспечение (исходный код
5. Результаты тестирования.
6. Вид формы приложения.
7. Выводы.

Лабораторная работа 3

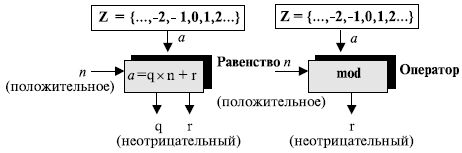
Этот материал необходим, чтобы подготовить студентов к дальнейшему разговору о криптографии. Рассматривается арифметика целых чисел, которая базируется на теории делимости и нахождении наибольшего общего делителя, обратить внимание на важность модульной арифметики (арифметики над вычетами по модулю n) и операций в ней, потому что они широко используются в криптографии.

### Модульная арифметика

В обычной жизни мы обычно пользуемся позиционной системой счисления. В позиционной системе счисления значение каждого числового знака (цифры) в записи числа зависит от его позиции (разряда) Однако существуют и так называемые «непозиционные системы счисления», к одной из которых относится «система остаточных классов» (СОК). Модулярная арифметика базируется на «Китайской теореме об остатках», которая для нашего случая звучит следующим образом: число X из диапазона [0; M), где M = p1\*p2\*…\*pn взаимо однозначно представимо в виде вектора (a1, a2, …, an), где ai = X mod pi (здесь и далее «mod» — операция взятия остатка от целочисленного деления X на pi).p1, … pn – модули системы a1, a2, …, an – остатки (вычеты) числа по заданной системе модулей.

Уравнение деления (a = q \times n + r), рассмотренное выше, имеет два входа (*a и n* ) и два выхода *( q и r* ). В модульной арифметике интерес представляет только один из выходов — остаток r. Мы не заботимся о частном q. Другими словами, когда мы делим a на n, мы интересуемся только тем, что *значение остатка равно* r. Это подразумевает, что мы можем представить изображение вышеупомянутого уравнения как *бинарный оператор* с двумя входами a и n и одним выходом r.

#### Операции по модулю

Вышеупомянутый *бинарный оператор* назван **оператором по модулю** и обозначается как mod. Второй вход ( n ) назван **модулем**. Вывод r назван **вычетом**. [Рисунок 1](http://www.intuit.ru/studies/courses/552/408/lecture/9351?page=4#image.2.9) показывает отношение деления по сравнению с оператором по модулю.

**Рис. 1.** Соотношение уравнения деления и оператора по модулю

Как показано на [рис.](http://www.intuit.ru/studies/courses/552/408/lecture/9351?page=4#image.2.9) 1, оператор по модулю (mod) выбирает целое число (a) из множества Z и положительный модуль (n). Оператор определяет неотрицательный остаток (r), что записывается в виде:

**a mod n = r**

**Пример 2.14**

Найти результат следующих операций:

a. **27 mod 5**

**b. 36 mod 12**

**c. –18 mod 14**

**d. –7 mod 10**

**Решение**

Мы ищем вычет r. Мы можем разделить a на n и найти q и r. Далее можно игнорировать q и сохранить r.

а. Разделим 27 на 5 - результат: r = 2. Это означает, что 27 mod 5 = 2.

б. Разделим 36 на 12 — результат: r = 0. Это означает, что 36 mod 12 = 0.

в. Разделим (–18) на 14 — результат: r = –4. Однако мы должны прибавить модуль (14), чтобы сделать остаток неотрицательным. Мы имеем r = –4 + 14 = 10. Это означает, что –18 mod 14 = 10.

г. Разделим (–7) на 10 — результат: r = –7. После добавления модуля –7 мы имеем r = 3. Это означает, что –7 mod 10 = 3.

#### 

#### Система вычетов: Zn

Результат операции по модулю n — всегда целое число между 0 и n - 1. Другими словами, результат a mod n — всегда неотрицательное целое число, меньшее, чем n. Мы можем сказать, что операция по модулю создает набор, который в модульной арифметике можно понимать как **систему наименьших вычетов по модулю n**, или Zn. Однако мы должны помнить, что хотя существует только одно множество целых чисел (Z), мы имеем бесконечное число множеств вычетов (Zn), но лишь одно для каждого значения n. Рисунок 2. показывает множество Zn и три множества Z2, Z6 и Z11.

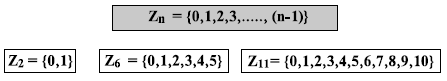


Рис. 2. Некоторые наборы Zn

#### 

#### Сравнения

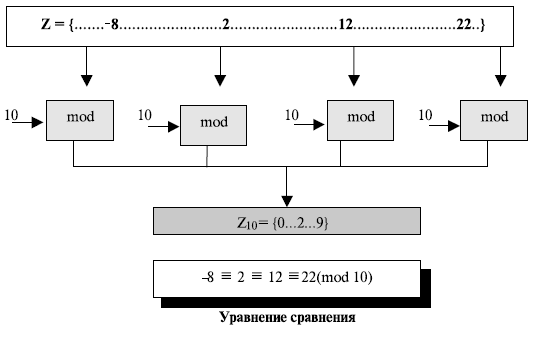
В криптографии мы часто используем понятие **сравнения** вместо равенства. Отображение Z в Zn не отображаются "один в один". Бесконечные элементы множества Z могут быть отображены одним элементом Zn. Например, результат 2 mod 10 = 2, 12 mod 10 = 2, 22 mod 10 = 2, и так далее. В модульной арифметике целые числа, подобные 2, 12, и 22, называются сравнимыми по модулю10 (mod 10). Для того чтобы указать, что два целых числа сравнимы, мы используем **оператор сравнения** ( \equiv  ). Мы добавляем mod n к правой стороне сравнения, чтобы определить значение модуля и сделать равенство правильным. Например, мы пишем:

с2 \equiv  12 (mod \ 10)   \ \    13 \equiv  23 (mod \ 10)  \ \   34 \equiv  24 (mod \ 10) \ \     –8 \equiv  12 (mod \ 10)
\\
3 \equiv  8 (mod \ 5)  \ \       8 \equiv  13 (mod \ 5)   \ \     23 \equiv  33 (mod \ 5)   \ \    -8 \equiv  2 (mod \ 5)мм

Рисунок 3. показывает принцип сравнения.

Мы должны объяснить несколько положений.

a. Оператор сравнения напоминает оператор равенства, но между ними есть различия. Первое: оператор *равенства* отображает элемент Z самого на себя; оператор *сравнения* отображает элемент Z на элемент Zn. Второе: оператор равенства показывает, что наборы слева и справа соответствуют друг другу "один в один", оператор сравнения — "многие — одному".



**Рис. 3.** Принцип сравнения

б. Обозначение (mod n), которое мы вставляем с правой стороны оператора сравнения, обозначает признак множества ( Zn ). Мы должны добавить это обозначение, чтобы показать, какой модуль используется в отображении. Символ, используемый здесь, не имеет того же самого значения, как бинарный оператор в уравнении деления. Другими словами, символ mod в выражении 12 mod 10 — оператор; а сочетание ( mod 10 ) в сравнении 2 \equiv 12(\bmod 10) означает, что набор — Z10.

#### 

#### Система вычетов

*Система вычетов* [a], или [a]n, — множество целых чисел, сравнимых по модулю n. Другими словами, это набор всех целых чисел, таких, что x = a (mod n). Например, если n = 5, мы имеем множество из пяти элементов [0], [1], [2], [3] и [4], таких как это показано ниже:

**[0] = {…., –15, -10, –5, 0, 5, 10, 15, …}**

**[1] = {…., –14, –9, –4, 1, 6 , 11, 16,…}**

**[2] = {…., –13, –8, –3, 2, 7, 12, 17,…}**

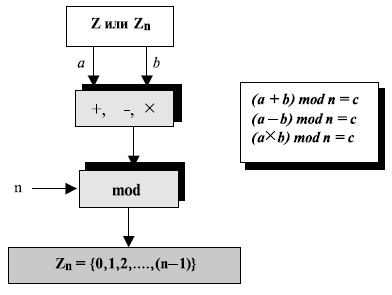
**[3] = {...., –12, –7, –2, 3, 8, 13, 18,…}**

**[4] = {…., –11, –6, –1, 4, 9, 14, 19,…}**

Целые числа в наборе [0] все дают остаток 0 при делении на 5 (сравнимы по модулю 5 ). Целые числа в наборе [1] все дают остаток 1 при делении на 5 (сравнимы по модулю 5 ), и так далее. В каждом наборе есть один элемент, называемый наименьшим (неотрицательным) вычетом. В наборе [0] это элемент 0 ; в наборе [1] — 1, и так далее. Набор, который показывает все наименьшие вычеты: Z5= {0, 1, 2, 3, 4}. Другими словами, набор Zn — набор

#### Операции в Zn

Три *бинарных операции* (*сложение, вычитание* и *умножение* ), которые мы обсуждали для Z, могут также быть определены для набора Zn. Результат, возможно, должен быть отображен в Zn с использованием операции по модулю, как это показано на [рис. 2.13](http://www.intuit.ru/studies/courses/552/408/lecture/9351?page=4#image.2.13).



**Рис. 4.** Бинарные операции в Zn

Фактически применяются два набора операторов: первый набор — один из *бинарных операторов* ( + ,-, \times ) ; второй — операторы по модулю. Мы должны использовать круглые скобки, чтобы подчеркнуть порядок работ. Как показано на [рис.](http://www.intuit.ru/studies/courses/552/408/lecture/9351?page=4#image.2.13) 4, входы ( a и b ) могут быть членами Z или Zn.

**Пример 2.16**

Выполните следующие операторы (поступающие от Zn ):

а. Сложение 7 и 14 в Z15

б. Вычитание 11 из 7 в Z13

в. Умножение 11 на 7 в Z20

**Решение**

Ниже показаны два шага для каждой операции:

**(14+7) mod 15 -> (21) mod 15 = 6**

**(7–11) mod 13 -> (-4) mod 13 = 9**

**(7x11) mod 20 -> (77) mod 20 = 17**

**Пример 2.17**

Выполните следующие операции (поступающие от Zn):

a. Сложение 17 и 27 в Z14

b. Вычитание 43 из 12 в Z13

c. Умножение 123 на -10 в Z19

**Решение**

Ниже показаны два шага для каждой операции:

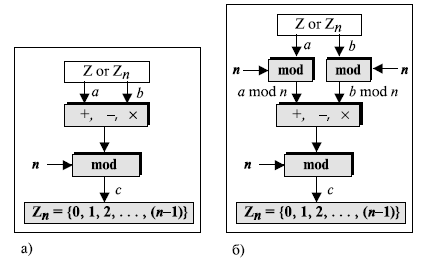
**(17 + 27) mod 14 -> (44) mod 14 = 2**

**(12 – 43) mod 13 -> (–31) mod 13 = 8**

**((123) x (–10)) mod 19 -> (–1230) mod 19 = 5**

##### Свойства

Мы уже упоминали, что два входа для трех *бинарных операторов* в сравнении по модулю могут использовать данные из Z или Zn. Следующие свойства позволяют нам сначала отображать два входа к Zn(если они прибывают от Z) перед выполнением этих трех *бинарных операторов*



**Рис. 5.** Свойства оператора mod

**Первое свойство: (a + b) mod n = [(a mod n) + (b mod n)] mod n**

**Второе свойство**: **(a – b) mod n = [(a mod n) - (b mod n)] mod n**

**Третье свойство**: **(a x b) mod n = [(a mod n) x (b mod n)] mod n**

Рисунок 5 показывает процесс до и после применения указанных выше свойств. Хотя по рисунку видно, что процесс с применением этих свойств более длинен, мы должны помнить, что в криптографии мы имеем дело с очень большими целыми числами. Например, если мы умножаем очень большое целое число на другое очень большое целое число, которое настолько большое, что не может быть записано в компьютере, то применение вышеупомянутых свойств позволяет уменьшить первые два операнда прежде, чем начать умножение. Другими словами, перечисленные свойства позволяют нам работать с меньшими числами.

**Пример 2.18**

Следующие примеры показывают приложение вышеупомянутых свойств.

1. \left( {1723345 + 2124945} \right)\bmod 11 = \left( {8 + 9} \right)\bmod 11 = 6
2. \left( {1723345 - 2124945} \right)\bmod 11 = \left( {8 - 9} \right)\bmod 11 = 10
3. \left( {1723345 \times 2124945} \right)\bmod 11 = \left( {8 \times 9} \right)\bmod 11 = 6

**Пример 2.19**

В арифметике мы часто должны находить остаток от степеней числа 10 при делении на целое число. Например, мы должны найти 10 mod 3, 102 mod 3, 103 mod 3, и так далее. Мы также должны найти 10 mod 7, 102 mod 7, 103 mod 7, и так далее. Третье свойство модульных операторов, упомянутое выше, делает жизнь намного проще.

**10n mod x = (10 mod x)n** Применение третьего свойства n раз.

Мы имеем

**10 mod 3 = 1 -> 10n mod 3 = (10 mod 3)n = 1**

**10 mod 9 = 1 -> 10n mod 9 = (10 mod 9)n = 1**

**10 mod 7 = 3 -> 10n mod 7 = (10 mod 7)n = 3n mod 7**

**Пример 2.20**

Мы уже говорили, что ***в арифметике остаток от целого числа, разделенного на 3, такой же, как остаток от суммы деления его десятичных цифр.*** Другими словами, остаток от деления 6371 равен остатку от деления суммы его цифр (17), на 3. Мы можем доказать, что это утверждение использует свойства модульного оператора. Запишем целое число как сумму его цифр, умноженных на степени 10.

**a = an10n +………+ a1101 + a0100**

**Например: 6371 = 6 x 103 + 3 x 102+ 7 x 101+ 1 x 100**

Теперь мы можем применить модульную операцию к двум сторонам равенства и использовать результат предыдущего примера, где **остаток 10n mod 3 равен 1.**

**a mod 3 = (an x 10n +…+ a1 x 101+ a0 x 100) mod 3**

**= (an x 10n) mod 3 +…+ (a1 x 101) mod 3 + (a0 x 100 mod 3) mod 3**

**= (an mod 3) x (10n mod 3) +…+ (a1 mod 3) x (101 mod 3) +**

**(a0 mod 3) x (100 mod 3) mod 3**

**= ((an mod 3) +…+ (a1 mod 3) + (a0 mod 3)) mod 3**

**= (an +…+ a1 + a0) mod 3**

На первый взгляд непонятно какое преимущество может дать такая система, однако существует 2 свойства, которые позволяют эффективно использовать модулярную арифметику в некоторых областях микроэлектроники:

1. Отсутствие переноса разрядов в сложении и умножении. Пусть нам дано два числа X1 и X2, представленные в виде системы остатков (x11, x12, …, x1n) и (x21, x22, …, x2n) по системе взаимно простых чисел (p1, p2, …, pn). В этом случае:X1 + X2 = ((x11+x21)modp1, (x12+x22)modp2, …, (x1n+x2n)modpn) X4 = X1\* X2= ((x11\*x21)modp1, (x12\*x22)modp2, …, (x1n\*x2n) mod pn) То есть что бы сложить или умножить два числа, достаточно сложить или умножить соответствующие элементы вектора, что для микроэлектроники означает, что это можно сделать параллельно и из-за малых размерностей p1, p2, …, pn сделать очень быстро.
2. Ошибка в одной позиции вектора не влияет на расчеты в других позициях вектора. В отличие от позиционной системы счисления все элементы вектора равнозначны и ошибка в одном из них ведет всего лишь к сокращению динамического диапазона. Этот факт позволяет проектировать устройства с повышенной отказоустойчивостью и коррекцией ошибок.
3. Вычисление mod *n* часто используется в криптографии, так как вычисление дискретных логарифмов и квадратных корней mod *n* может быть нелегкой проблемой. Арифметика вычетов, к тому же, легче реализуется на компьютерах, поскольку она ограничивает диапазон промежуточных значений и результата. Для k-битовых вы­четов n, промежуточные результаты любого сложения, вычитание или умножения будут не длиннее, чем 2 *к* бит. Поэтому в арифметике вычетов мы можем выполнить возведение в степень без огромных промежуточных результатов. Вычисление степени некоторого числа по модулю другого числа,

*a* mod *n,*

представляет собой просто последовательность умножений и делений, но существуют приемы, ускоряющие это действие. Один из таких приемов стремится минимизировать количество умножений по модулю, другой - оптимизировать отдельные умножения по модулю. Так как операции дистрибутивны, быстрее выполнить возведение в степень как поток последовательных умножений, каждый раз получая вычеты. Сейчас вы не чувствуете разницы, но она будет заметна при умножении 200-битовых чисел.

Например, если вы хотите вычислить *a7* mod *n,* не выполняйте наивно семь умножений и одно приведение по модулю:

*(а\*а\*а\*а\*а\*а\*а\*а)* mod n

Вместо этого выполните три меньших умножения и три меньших приведения по модулю:

*((a2* mod *п)2* mod *n)2* mod *n*

Точно также,

a16 mod *n* =(((n2 mod *n)2* mod *n)2* mod *n)2* mod *n*

Вычисление *аx,* где *х* не является степенью 2, ненамного труднее. Двоичная запись представляет *х* в виде суммы степеней 2: 25 - это бинарное 11001, поэтому 25 = 24 + 23 + 20. Поэтому

*a25* mod *п = (а\*а24)* mod *n = (а\*а8\*а16)* mod *n =*

*= (а\*(( а2)2)* 2\*((( *а1)*2)2)2) mod *п* = (а\*((( *а\*а2)*2)2)2) mod *n*

С продуманным сохранением промежуточных результатов вам понадобится только шесть умножений:

(((((((a2 mod n)\* *a)2* mod *n)2* mod n)2 mod *n)2* mod *n)2 \*a)* mod *n*

Такой прием называется **цепочкой сложений,** или методом двоичных квадратов и умножения. Он использует простую и очевидную цепочку сложений, в основе которой лежит двоичное представление числа.

***Обратные значения по модулю***

Помните, что такое обратные значения? Обратное значение для 4 - 1/4, потому что 4\*1/4 =1. В мире вычетов проблема усложняется:

*4\*х =* 1 (mod 7)

Это уравнение эквивалентно обнаружению х и к, таких что

***Ах*** *= 7\*k+1\*

где *х и к -* целые числа. Общая задача состоит в нахождении х, такого что

1 = *(а\*х)* mod *n*

Это также можно записать как

*a*-1 = *х* (mod *n*)

Проблему обратных значений по модулю решить нелегко. Иногда у нее есть решение, иногда нет. Например, обратное значение 5 по модулю 14 равно 3. С другой стороны у числа 2 нет обратного значения по модулю 14.

В общем случае у уравнения *a-1 = х (*mod *n)* существует единственное решение, если *а и п* взаимно просты. Если *а* и *n* не являются взаимно простыми, то *а-1* = *х* (mod *п)* не имеет решений. Если *n* является простым чис-лом, то любое число от 1 до *n-1* взаимно просто с *п* и имеет в точности одно обратное значение по модулю *п.* Как искать обратное значение *a*  по модулю *n?* Обратное значение *а* по модулю *п* можно вычислить с помощью алгоритма Эвклида. Иногда это называется расширенным алгоритмом Эвклида.

Задание на практическое выполнение работы:

1.Изучить теоретические положения модульной арифметики.

2.Разработать визуальное приложение (алгоритм и программное обеспечение):

* Расчет вычетов;
* по заданному модулю *n* построить ряд сравнения;
* реализация свойств трех бинарных операций (сложение, вычитание, умножение);
* нахождение обратного значения числа;
* возведение в степень.

В алгоритмах шифрования достаточно часто используется операция в степень по модулю натурального числа *ad (mod n)*. Операцию возведения в степень выполнять по алгоритму:

1. число *d* представить в двоичной системе счисления

d=d0\*2m +…+dm-1\*2 + dm . I =0,m

di – цифры в двоичном представлении, равные 0 или 1. d0 = 1.

1. Положить а0 = а, затем для i = 1, …. m вычислить

ai = (ai-1)2 \*adi (mod n)

1. am есть искомое число ad (mod n)

Должна быть обеспечена наглядность выполнения приложения. Для создания программного обеспечения и подтверждения его работоспособности тестирование проводить не менее чем на 10 различных наборах данных.

В отчете по лабораторной работе должно содержаться:

1. Титульный лист в соответствии со стандартом ТПУ.

2. Задание на лабораторную работу.

3. Краткую теорию.

4. Ход выполнения работы: алгоритмы, программное обеспечение (исходный код), результаты тестирования.

5. Вывод

### Лабораторная работа 4

### Шифр Цезаря

Шифр**Цезаря** — один из древнейших шифров. При шифровании каждый символ заменяется другим, отстоящим от него в алфавите на фиксированное число позиций. Шифр Цезаря можно классифицировать как шифр подстановки, при более узкой классификации — шифр простой замены. Шифр назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки. Естественным развитием шифра Цезаря стал шифр Виженера. С точки зрения современного криптоанализа, шифр Цезаря не имеет приемлемой стойкости.

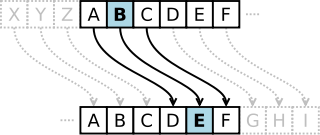


Рисунок 1 Шифр Цезаря 

#### Математическая модель

Если сопоставить каждому символу алфавита его порядковый номер (нумеруя с 0), то шифрование и дешифрование можно выразить формулами:

http://kriptografea.narod.ru/13.png

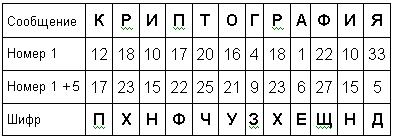
http://kriptografea.narod.ru/14.png

где x — символ открытого текста   
y — символ шифрованного текста   
n — мощность алфавита (кол-во символов)   
k — ключ.   
Можно заметить, что суперпозиция двух шифрований на ключах k1 и k2 — есть просто шифрование на ключе k1+k2. Более общее, множество шифрующих преобразований шифра Цезаря образует группу Z.

Алфавит:



**Пример:**



Задание на выполнение работы

1. Разработать криптографическое преобразование текста по алгоритму

Цезаря.

2. Подготовить данные для шифрования текста (алфавит).

3. Разработать приложение, показывающее процесс криптографического преобразования текста. Приложение должно содержать:

* Алфавит с обозначением позиции букв;
* Окно исходного текста;
* Окно для ввода значения ключа;
* Кнопки управления процессами шифрования и расшифрования;
* Окна результата работы приложения;
* Кнопка завершения работы приложения.

4. Длину текста брать не менее 10-15 символов.

5. Тестирование работы приложения выполнить при разных значениях ключа (не менее трех).

6. Результаты работы приложения записать в файл.

В отчете по лабораторной работе должно содержаться:

1. Титульный лист в соответствии со стандартом ТПУ.

2. Задание на лабораторную работу.

3. Краткую теорию.

4. Ход выполнения работы: алгоритмы, программное обеспечение (исходный код), результаты тестирования.

5. Вывод.

**Лабораторная работа 4**

Парольная защита

Под **несанкционированным доступом к информации** (НСД) согласно руководящим документам Гостехкомиссии будем понимать доступ к информации, нарушающий установленные правила разграничения доступа и осуществляемый с использованием штатных средств, предоставляемых СВТ или АС. НСД может носить случайный или намеренный характер. Можно выделить несколько обобщенных категорий методов защиты от НСД, в частности:

* организационные;
* технологические;
* правовые.

К первой категории относятся меры и мероприятия, регламентируемые внутренними инструкциями организации, эксплуатирующей информационную систему. Пример такой защиты — присвоение грифов секретности документам и материалам, хранящимся в отдельном помещении, и контроль доступа к ним сотрудников. Вторую категорию составляют механизмы защиты, реализуемые на базе программно-аппаратных средств, например систем идентификации и аутентификации или охранной сигнализации. Последняя категория включает меры контроля за исполнением нормативных актов общегосударственного значения, механизмы разработки и совершенствования нормативной базы, регулирующей вопросы защиты информации. Реализуемые на практике методы, как правило, сочетают в себе элементы нескольких из перечисленных категорий. Так, управление доступом в помещения может представлять собой взаимосвязь организационных (выдача допусков и ключей) и технологических (установку замков и систем сигнализации) способов защиты.

Рассмотрим подробнее такие взаимосвязанные методы защиты от НСД, как идентификация, аутентификация и используемое при их реализации криптографическое преобразование информации.

**Идентификация** — это присвоение пользователям идентификаторов и проверка предъявляемых идентификаторов по списку присвоенных.

**Аутентификация** — это проверка принадлежности пользователю предъявленного им идентификатора. Часто аутентификацию также называют подтверждением или проверкой подлинности.

Под безопасностью (стойкостью) системы идентификации и аутентификации будем понимать степень обеспечиваемых ею гарантий того, что злоумышленник не способен пройти аутентификацию от имени другого пользователя, В этом смысле, чем выше стойкость системы аутентификации, тем сложнее злоумышленнику решить указанную задачу. Система идентификации и аутентификации является одним из ключевых элементов инфраструктуры защиты от НСД любой информационной системы.

Различают три группы методов аутентификации, основанных на наличии у каждого пользователя:

* индивидуального объекта заданного типа;
* знаний некоторой известной только ему и проверяющей стороне информации;
* индивидуальных биометрических характеристик.

К первой группе относятся методы аутентификации, использующие удостоверения, пропуска, магнитные карты и другие носимые устройства, которые широко применяются для контроля доступа в помещения, а также входят в состав программно-аппаратных комплексов защиты от НСД к средствам вычислительной техники.

Во вторую группу входят методы аутентификации, использующие пароли. По экономическим причинам они включаются в качестве базовых средств защиты во многие программно-аппаратные комплексы защиты информации. Все современные операционные системы и многие приложения имеют встроенные механизмы парольной защиты.

Последнюю группу составляют методы аутентификации, основанные на применении оборудования для измерения и сравнения с эталоном заданных индивидуальных характеристик пользователя: тембра голоса, отпечатков пальцев, структуры радужной оболочки глаза и др. Такие средства позволяют с высокой точностью аутентифицировать обладателя конкретного биометрического признака, причем "подделать" биометрические параметры практически невозможно. Однако широкое распространение подобных технологий сдерживается высокой стоимостью необходимого оборудования.

Если в процедуре аутентификации участвуют только две стороны, устанавливающие подлинность друг друга, такая процедура называется непосредственной аутентификацией (direct password authentication). Если же в процессе аутентификации участвуют не только эти стороны, но и другие, вспомогательные, говорят об аутентификации с участием доверенной стороны (trusted third party authentication). При этом третью сторону называют сервером аутентификации (authentication server) или арбитром (arbitrator).

Наиболее распространенные методы аутентификации основаны на применении многоразовых или одноразовых паролей. Из-за своего широкого распространения и простоты реализации парольные схемы часто в первую очередь становятся мишенью атак злоумышленников. Эти методы включают следующие разновидности способов аутентификации:

* по хранимой копии пароля или его свёртке (plaintext-equivalent);
* по некоторому проверочному значению (verifier-based);
* без непосредственной передачи информации о пароле проверяющей стороне (zero-  
  knowledge);
* с использованием пароля для получения криптографического ключа (cryptographic).

В первую разновидность способов входят системы аутентификации, предполагающие наличие у обеих сторон копии пароля или его свертки. Для организации таких систем требуется создать и поддерживать базу данных, содержащую пароли или сверки паролей всех пользователей. Их слабой стороной является то, что получение злоумышленником этой базы данных позволяет ему проходить аутентификацию от имени любого пользователя.

Способы, составляющие вторую разновидность, обеспечивают более высокую степень безопасности парольной системы, так как проверочные значения, хотя они и зависят от паролей, не могут быть непосредственно использованы злоумышленником для аутентификации.

Наконец, аутентификация без предоставления проверяющей стороне какой бы то ни было информации о пароле обеспечивает наибольшую степень защиты. Этот способ гарантирует безопасность даже в том случае, если нарушена работа проверяющей стороны (например, в программу регистрации в системе внедрен "троянский конь").

Особым подходом в технологии проверки подлинности являются криптографические протоколы аутентификации. Такие протоколы описывают последовательность действий, которую должны совершить стороны для взаимной аутентификации, кроме того, эти действия, как правило, сочетаются с генерацией и распределением криптографических ключей для шифрования последующего информационного обмена. Корректность протоколов аутентификации вытекает из свойств задействованных в них математических и криптографических преобразований и может быть строго доказана.

Обычные парольные системы проще и дешевле для реализации, но менее безопасны, чем системы с криптографическими протоколами. Последние обеспечивают более надежную защиту и дополнительно решают задачу распределения ключей. Однако используемые в них технологии могут быть объектом законодательных ограничений. Для более детального рассмотрения принципов построения парольных систем сформулируем несколько основных определений.

**Идентификатор пользователя** — некоторое уникальное количество информации, позволяющее различать индивидуальных пользователей парольной системы (проводить их идентификацию). Часто идентификатор также называют именем пользователя или именем учетной записи пользователя.

**Пароль пользователя** — некоторое секретное количество информации, известное только пользователю и парольной системе, которое может быть запомнено пользователем и предъявлено для прохождения процедуры аутентификации. Одноразовый пароль дает возможность пользователю однократно пройти аутентификацию. Многоразовый пароль может быть использован для проверки подлинности повторно.

**Учетная запись пользователя** — совокупность его идентификатора и его пароля.

База данных пользователей парольной системы содержит учетные записи всех пользователей данной парольной системы.

Под **парольной системой** будем понимать программно-аппаратный комплекс, реализующий системы идентификации и аутентификации пользователей АС на основе одноразовых или многоразовых паролей. Как правило, такой комплекс функционирует совместно с подсистемами разграничения доступа и регистрации событий. В отдельных случаях парольная система может выполнять ряд дополнительных функций, в частности генерацию и распределение кратковременных (сеансовых) криптографических ключей.

Основными компонентами парольной системы являются:

* интерфейс пользователя;
* интерфейс администратора;
* модуль сопряжения с другими подсистемами безопасности;
* база данных учетных записей.

Парольная система представляет собой "передний край обороны" всей системы безопасности. Некоторые ее элементы (в частности, реализующие интерфейс пользователя) могут быть расположены в местах, открытых для доступа потенциальному злоумышленнику. Поэтому парольная система становится одним из первых объектов атаки при вторжении злоумышленника в защищенную систему. Ниже перечислены типы угроз безопасности парольных систем:

1. Разглашение параметров учетной записи через:

* подбор в интерактивном режиме;
* подсматривание;
* преднамеренную передачу пароля его владельцем другому лицу;
* захват базы данных парольной системы (если пароли не хранятся в базе в  
  открытом виде, для их восстановления может потребоваться подбор или  
  дешифрование);
* перехват переданной по сети информации о пароле;
* хранение пароля в доступном месте.

2. Вмешательство в функционирование компонентов парольной системы через:

* внедрение программных закладок;
* обнаружение и использование ошибок, допущенных на стадии разработки;
* выведение из строя парольной системы.

Некоторые из перечисленных типов угроз связаны с наличием так называемого человеческого фактора, проявляющегося в том, что пользователь может:

* выбрать пароль, который легко запомнить и также легко подобрать;
* записать пароль, который сложно запомнить, и положить запись в доступном месте;

Обычные парольные системы проще и дешевле для реализации, но менее безопасны, чем системы с криптографическими протоколами. Последние обеспечивают более надежную защиту и дополнительно решают задачу распределения ключей. Однако используемые в них технологии могут быть объектом законодательных ограничений. Для более детального рассмотрения принципов построения парольных систем сформулируем несколько основных определений.

**Идентификатор пользователя** — некоторое уникальное количество информации, позволяющее различать индивидуальных пользователей парольной системы (проводить их идентификацию). Часто идентификатор также называют именем пользователя или именем учетной записи пользователя.

**Пароль пользователя** — некоторое секретное количество информации, известное только пользователю и парольной системе, которое может быть запомнено пользователем и предъявлено для прохождения процедуры аутентификации. Одноразовый пароль дает возможность пользователю однократно пройти аутентификацию. Многоразовый пароль может быть использован для проверки подлинности повторно.

**Учетная запись пользователя** — совокупность его идентификатора и его пароля.

База данных пользователей парольной системы содержит учетные записи всех пользователей данной парольной системы.

Под **парольной системой** будем понимать программно-аппаратный комплекс, реализующий системы идентификации и аутентификации пользователей АС на основе одноразовых или многоразовых паролей. Как правило, такой комплекс функционирует совместно с подсистемами разграничения доступа и регистрации событий. В отдельных случаях парольная система может выполнять ряд дополнительных функций, в частности генерацию и распределение кратковременных (сеансовых) криптографических ключей.

Основными компонентами парольной системы являются:

* интерфейс пользователя;
* интерфейс администратора;
* модуль сопряжения с другими подсистемами безопасности;
* база данных учетных записей.

Парольная система представляет собой "передний край обороны" всей системы безопасности. Некоторые ее элементы (в частности, реализующие интерфейс пользователя) могут быть расположены в местах, открытых для доступа потенциальному злоумышленнику. Поэтому парольная система становится одним из первых объектов атаки при вторжении злоумышленника в защищенную систему. Ниже перечислены типы угроз безопасности парольных систем:

1. Разглашение параметров учетной записи через:

* подбор в интерактивном режиме;
* подсматривание;
* преднамеренную передачу пароля его владельцем другому лицу;
* захват базы данных парольной системы (если пароли не хранятся в базе в  
  открытом виде, для их восстановления может потребоваться подбор или  
  дешифрование);
* перехват переданной по сети информации о пароле;
* хранение пароля в доступном месте.

2. Вмешательство в функционирование компонентов парольной системы через:

* внедрение программных закладок;
* обнаружение и использование ошибок, допущенных на стадии разработки;
* выведение из строя парольной системы.

Некоторые из перечисленных типов угроз связаны с наличием так называемого человеческого фактора, проявляющегося в том, что пользователь может:

* выбрать пароль, который легко запомнить и также легко подобрать;
* записать пароль, который сложно запомнить, и положить запись в доступном месте;

Обычные парольные системы проще и дешевле для реализации, но менее безопасны, чем системы с криптографическими протоколами. Последние обеспечивают более надежную защиту и дополнительно решают задачу распределения ключей. Однако используемые в них технологии могут быть объектом законодательных ограничений. Для более детального рассмотрения принципов построения парольных систем сформулируем несколько основных определений.

**Идентификатор пользователя** — некоторое уникальное количество информации, позволяющее различать индивидуальных пользователей парольной системы (проводить их идентификацию). Часто идентификатор также называют именем пользователя или именем учетной записи пользователя.

**Пароль пользователя** — некоторое секретное количество информации, известное только пользователю и парольной системе, которое может быть запомнено пользователем и предъявлено для прохождения процедуры аутентификации. Одноразовый пароль дает возможность пользователю однократно пройти аутентификацию. Многоразовый пароль может быть использован для проверки подлинности повторно.

**Учетная запись пользователя** — совокупность его идентификатора и его пароля.

База данных пользователей парольной системы содержит учетные записи всех пользователей данной парольной системы.

Под **парольной системой** будем понимать программно-аппаратный комплекс, реализующий системы идентификации и аутентификации пользователей АС на основе одноразовых или многоразовых паролей. Как правило, такой комплекс функционирует совместно с подсистемами разграничения доступа и регистрации событий. В отдельных случаях парольная система может выполнять ряд дополнительных функций, в частности генерацию и распределение кратковременных (сеансовых) криптографических ключей.

Основными компонентами парольной системы являются:

* интерфейс пользователя;
* интерфейс администратора;
* модуль сопряжения с другими подсистемами безопасности;
* база данных учетных записей.

Парольная система представляет собой "передний край обороны" всей системы безопасности. Некоторые ее элементы (в частности, реализующие интерфейс пользователя) могут быть расположены в местах, открытых для доступа потенциальному злоумышленнику. Поэтому парольная система становится одним из первых объектов атаки при вторжении злоумышленника в защищенную систему. Ниже перечислены типы угроз безопасности парольных систем:

1. Разглашение параметров учетной записи через:

* подбор в интерактивном режиме;
* подсматривание;
* преднамеренную передачу пароля его владельцем другому лицу;
* захват базы данных парольной системы (если пароли не хранятся в базе в  
  открытом виде, для их восстановления может потребоваться подбор или  
  дешифрование);
* перехват переданной по сети информации о пароле;
* хранение пароля в доступном месте.

2. Вмешательство в функционирование компонентов парольной системы через:

* внедрение программных закладок;
* обнаружение и использование ошибок, допущенных на стадии разработки;
* выведение из строя парольной системы.

Некоторые из перечисленных типов угроз связаны с наличием так называемого человеческого фактора, проявляющегося в том, что пользователь может:

* выбрать пароль, который легко запомнить и также легко подобрать;
* записать пароль, который сложно запомнить, и положить запись в доступном месте;
* ввести пароль так, что его смогут увидеть посторонние:
* передать пароль другому лицу намеренно или под влиянием заблуждения.

В дополнение к выше сказанному необходимо отметить существование "парадокса человеческого фактора". Заключается он в том, что пользователь нередко стремится выступать скорее противником парольной системы, как, впрочем, и любой системы безопасности, функционирование которой влияет на его рабочие условия, нежели союзником системы защиты, тем самым ослабляя ее. Защита от указанных угроз основывается на ряде перечисленных ниже организационно-технических мер и мероприятий.

Выбор паролей

В большинстве систем пользователи имеют возможность самостоятельно выбирать пароли или получают их от системных администраторов. При этом для уменьшения деструктивного влияния описанного выше человеческого фактора необходимо реализовать ряд требований к выбору и использованию паролей.

Таблица

|  |  |
| --- | --- |
| **Требование к выбору пароля** | **Получаемый эффект** |
| Установление минимальной длины пароля | Усложняет задачу злоумышленника при попытке подсмотреть пароль или подобрать пароль методом «тотального опробования» |
| Использование в пароле различных групп символов | Усложняет задачу злоумышленника при попытке подобрать пароль методом «тотального опробования» |
| Проверка и отбраковка пароля по словарю | Усложняет задачу злоумышленника при попытке подобрать пароль по словарю |
| Установление максимального срока действия пароля | Усложняет задачу злоумышленника при попытке подобрать пароль методом «тотального опробования», в том числе без непосредственного обращения к системе защиты (режим off-line) |
| Установление минимального срока действия пароля | Препятствует попыткам пользователя заменить пароль на старый после его смены по предыдущему требованию |
| Ведение журнала истории паролей | Обеспечивает дополнительную степень защиты по предыдущему требованию |
| Применение эвристического алгоритма, бракующего пароли на основании данных журнала истории | Усложняет задачу злоумышленника при попытке подобрать пароль по словарю или с использованием эвристического алгоритма |
| Ограничение числа попыток ввода пароля | Препятствует интерактивному подбору паролей злоумышленником |
| Поддержка режима принудительной смены пароля пользователя | Обеспечивает эффективность требования, ограничивающего максимальный срок действия пароля |
| Использование задержки при вводе неправильного пароля | Препятствует интерактивному подбору паролей злоумышленником |
| Запрет на выбор пароля самими пользователями и автоматическая генерация паролей | Исключает возможность подобрать пароль по словарю. Если алгоритм генерации паролей не известен злоумышленнику, последний может подбирать пароли только методом «тотального опробования» |
| Принудительная смена пароля при первой регистрации пользователя в системе | Защищает от неправомерных действия системного администратора, имеющего доступ к паролю в момент создания учетной записи |

**2. Примеры.**

1.Пример **1**.

Задание:

определить время перебора всех паролей, состоящих из 6 цифр.

Алфавит составляют цифры n=10.

Длина пароля 6 символов k=6.

Таким образом, получаем количество вариантов: С=nk= 106

Примем скорость перебора s=10 паролей в секунду. Получаем время перебора всех

паролей t= С/8=105 секунд1667 минут28 часов1,2 дня.



Примем, что после каждого из m=3 неправильно введенных паролей идет пауза в v=5 секунд. Получаем время перебора всех паролей

Т=t\*5/3=16667 секунд2778 минут46 часов1,9 дня.



Титог = t+Т= 1,2+1,9 = 3,1 дня

2. Пример 2.

Определить минимальную длину пароля, алфавит которого состоит из 10 символов, время перебора которого было не меньше 10 лет.

Алфавит составляют символы n=10.

Длина пароля рассчитывается: k=logn C= lg С.

Определим количество вариантов С= t \* s=10лет\*10 паролей в сек. =

10\*10\*365\*24\*60\*60\*3,15\* 109вариантов.

Таким образом, получаем длину пароля: k=lg (3,15\*109) = 9,5

Очевидно, что длина пароля должна быть не менее 10 символов.

**3. Задания.**

1. Определить время перебора всех паролей с параметрами.

Алфавит состоит из n символов.

Длина пароля символов k.

Скорость перебора s паролей в секунду.

После каждого из m неправильно введенных паролей идет пауза в v секунд

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| вариант | n | K | s | m | V |
| 1 | 33 | 10 | 100 | 0 | 0 |
| 2 | 26 | 12 | 13 | 3 | 2 |
| 3 | 52 | 6 | 30 | 5 | 10 |
| 4 | 66 | 7 | 20 | 10 | 3 |
| 5 | 59 | 5 | 200 | 0 | 0 |
| 6 | 118 | 9 | 50 | ' 7 | 12 |
| 7 | 128 | 10 | 500 | 0 | 0 |
| 8 | 150 | 3 | 200 | 5 | 3 |
| 9 | 250 | 8 | 600 | 7 | 3 |
| 10 | 500 | 5 | 1000 | 10 | 10 |

2. Определить минимальную длину пароля, алфавит которого состоит из n символов, время перебора которого было не меньше t лет. Скорость перебора s паролей в секунду.

|  |  |  |  |
| --- | --- | --- | --- |
| вариант | n | t | s |
| 1 | 33 | 100 | 100 |
| 2 | 26 | 120 | 13 |
| 3 | 52 | 60 | 30 |
| 4 | 66 | 70 | 20 |
| 5 | 59 | 50 | 200 |
| 6 | 118 | 90 | 50 |
| 7 | 128 | 100 | 500 |
| 8 | 150 | 30 | 200 |
| 9 | 250 | 80 | 600 |
| 10 | 500 | 50 | 1000 |

3. Определить количество символов алфавита, пароль состоит из k символов, время перебора которого было не меньше t лет. Скорость перебора s паролей в секунду.

|  |  |  |  |
| --- | --- | --- | --- |
| вариант | k | t | s |
| 1 | 5 | 100 | 100 |
| 2 | 6 | 120 | 13 |
| 3 | 10 | 60 | 30 |
| 4 | 7 | 70 | 20 |
| 5 | 9 | 50 | 200 |
| 6 | 11 | 90 | 50 |
| 7 | 12 | 100 | 500 |
| 8 | 6 | 30 | 200 |
| 9 | 8 | 80 | 600 |
| 10 | 50 | 50 | 1000 |

В отчете по лабораторной работе должно содержаться:

1. Титульный лист в соответствии со стандартом ТПУ.
2. Задание на лабораторную работу.
3. Краткую теорию.
4. Ход выполнения работы: алгоритмы, программное обеспечение (исходный код),  
   результаты тестирования.
5. Выводы.

**Лабораторная работа 4**

Обмен ключами по Диффи-Хелману

1. Теория.

Для защиты информации в вычислительных сетях используется такой Е криптографического преобразования как шифрование, в котором всегда различают два элемента: ключ и алгоритм. При этом ключом является секретное состояние некоторых параметров алгоритма криптопреобразования сообщения.

На практике в зависимости от способа применения ключа различают *i* типа криптографических систем:

* Одноключевые (симметричные)
* Двухключевые (несимметричные)

В одноключевых системах, называемых традиционными, ключи шифрования и расшифрования (л.р.2) либо одинаковы, либо легко выводятся один из другого, обеспечивая таким образом единый общий ключ. Такой ключ является секретным и передается получателю сообщения только по защищенному каналу связи.

Однако при этом имеет место следующий парадокс: если для обмена ее секретного ключа используется защищенный канал, то нет необходимости шифровать конфиденциальные сообщения, гораздо проще отправить их по это: каналу.

Отмеченный парадокс может быть исключен использованием идеи Диффи и Хеллмана, которые предложили способ выработки секретного ключа без предварительного согласования между абонентами сети путем обмена информацией по открытому каналу. Этот способ был предложен Диффи и Хеллманом в 1976 году.

Этот метод позволяет пользователям обмениваться ключами по незащищенным каналам связи. Его безопасность обусловлена трудностью вычисления дискретных логарифмов в конечном поле, в отличие от легкости решения прямой задачи дискреиного возведения в степень в том же конечном поле. Суть метода Диффи-Хеллмана заключается в следующем. (См. ниже рисунок).

Пользователи *А и В,* участвующие в обмене информацией, генерируют независимодруг от друга свои случайные секретные ключи *kA* и *kB* (ключи *kA* и *kB -случа*йные большие целые числа, которые хранятся пользователями *А* и *В* в секрете).

Затем пользователь *А* вычисляет на основании своего секретного ключа *kA откры*тый ключ



одновременно пользователь *В* вычисляет на основании своего секретного ключа *kB* открытый ключ



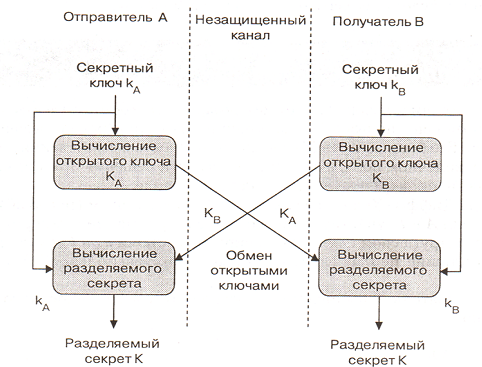


Рис. Генерация ключей по алгритму Диффи-Хелмана

где *N и g-* большие целые простые числа. Арифметические действия выполняются с приведением по модулю *N*. Числа *N и g* могут не храниться в секрете. Как правило, эти значения являются общими для всех пользователей сети или системы. Затем пользователи *А* и *В* обмениваются своими открытыми ключами *КA* и *КB* по незащищенному каналу и используют их для вычисления общего сессионного ключа *К* (разделяемого секрета):

* пользователь А:
* пользователь В: 
* при этом 

Таким образом, результатом этих действий оказывается общий сессионный ключ, который является функцией обоих секретных ключей - *kA* и *kB.*

Злоумышленник, перехвативший значения открытых ключей *КA* и *КB,* не мо­жет вычислить сессионный ключ *К,* потому что он не имеет соответствующих значений секретных ключей *кA* и *kB.* Благодаря использованию однонаправлен­ной функции операция вычисления открытого ключа необратима, то есть невозможно по значению открытого ключа абонента вычислить его секрет­ный ключ.

Уникальность метода Диффи-Хеллмана заключается в том, что пара абонентов имеет возможность получить известное только им секретное число, передавая по открытой сети открытые ключи. После этого абоненты могут приступить к за­щите передаваемой информации уже известным проверенным способом - при­меняя симметричное шифрование с использованием полученного разделяемого секрета.

Схема Диффи-Хеллмана дает возможность шифровать данные при каждом се­ансе связи на новых ключах. Это позволяет не хранить секреты на дискетах или других носителях. Не следует забывать, что любое хранение секретов повышает вероятность попадания их в руки конкурентов или противника.

Схема Диффи-Хеллмана позволяет реализовать *метод комплексной защиты конфиденциальности и аутентичности передаваемых данных.* Эта схема предос­тавляет пользователям возможность сформировать и использовать одни и те же ключи для выполнения цифровой подписи и симметричного шифрования переда­ваемых данных.

3. Практика.

Составьте программное обеспечение, реализующее алгоритм обмена ключами. Ключи должны автоматически формироваться в файлы. Должна быть обеспечена наглядность выполнения алгоритма. Для созданного программного обеспечения проведите тестирование не менее чем на 10 различных наборах данных.

**Лабораторная работа 5**

Шифр RSA

1. Теория.

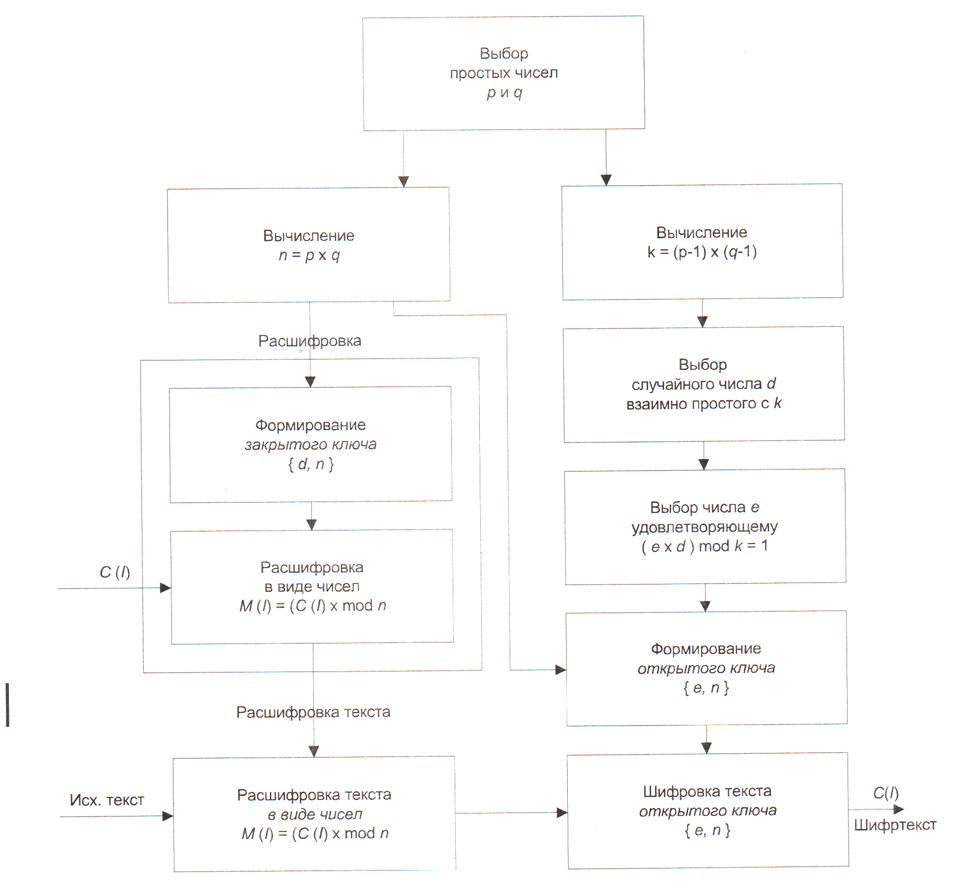
Защита данных с помощью криптографического преобразования является эффективным решением проблемы их безопасности. Зашифрованные данные доступны лишь тем, кто знает, как их расшифровать, то есть тем, кто обладает соответствующим ключом шифрования.

Одним из наиболее перспективных криптографических стандартов на шифрование данных являются системы с открытым ключом. В таких системах для шифрования используется один ключ, а для расшифрования другой. Первый ключ является открытым и может быть опубликован для шифрования своей информации любым пользователем сети. Получатель зашифрованной информации для расшифровки данных использует второй ключ, являющийся секретным. При этом должно соблюдаться следующее условие: секретный ключ не может быть определен из опубликованного открытого ключа.

Криптографические системы с открытым ключом используют необратимые или односторонние функции, обладающие важным свойством: при заданном значении *х* относительно просто вычислить значение *f(x),* однако, если\_у *=f(x),* то нет простого пути для вычисления значения х, то есть очень трудно рассчитать значение обратной функции *1/f(y).*

В настоящее время широко используется метод криптографической защиты данных с открытым ключом RSA, получившим название по начальным буквам фамилий его изобретателей (Rivest, Shamir, Adleman). На основе метода RSA разработаны алгоритмы шифрования, успешно применяемые для защиты информации. Он обладает высокой криптостойкостью и может быть реализован при использовании относительно несложных программных и аппаратных средств. Данный метод позволил решить проблему обеспечения персональных подписей в условиях безбумажной передачи и обработки данных. Описание схем формирования шифротекста в алгоритмах типа RSA приведено в различной литературе.

Использование метода RSA для криптографической защиты информации может быть пояснено с помощью структурной схемы, представленной на рисунке.



Надежность алгоритма RSA основывается на трудности факторизации больших чисел и сложности вычисления дискретных логарифмов в конечном поле.

В алгоритме RSА открытый ключ *КB,* секретный ключ *kB,* сообщение *М* и криптограмма *С* принадлежат множеству целых чисел

*ZN={0,1,2,...,N-* 1},

где *N* - модуль: *N=P \* Q*

Здесь *Р* и *Q -* случайные большие *простые* числа. Для обеспечения макксимальной безопасности выбирают *Р и Q* равной длины и хранят в секрете.

Открытый ключ *КB* выбирают случайным образом так, чтобы выполнялись следующие условия:

*1<КВ<φ(N)*, НОД *(Кв, φ(N))=1*

*φ(N)=(P-1)(Q-1),*

где *φ(N)* - функция Эйлера.

Функция Эйлера *φ(N)* указывает количество положительных целых чисел в интервале от 1 до *N,* которые взаимно просты с *N.*

Второе из указанных выше условий означает, что открытый ключ *КB* и функция Эйлера *φ(N*)должны быть взаимно простыми.

Далее, используя расширенный алгоритм Евклида, вычисляют секретный ключ *kB,* такой, что

*kB\*KB =* l (mod *φ(N))*

*или*

*kB=KB (mod(P-l)(Q-l)).*

Это можно осуществить, так как получатель *В* знает пару простых чисел (Р, *Q)* и может легко найти *φ(N*). Заметим, что *kB* и *N* должны быть взаимно простыми.

Открытый ключ *КB* используют для шифрования данных, а секретный ключ *kB -*для расшифрования.

Процедура шифрования определяет криптограмму С через пару (открытый ключ *КB,* сообщение *М*) в соответствии со следующей формулой:

*С *

В качестве алгоритма быстрого вычисления значения *С* используют ряд последовательных возведений в квадрат целого *М* и умножений на *М* с приведением по модулю *N.*

Расшифрование криптограммы С выполняют, используя пару (секретный ключ *k~~B~~* криптограмма С) по следующей формуле:



***Процедуры шифрования и расшифрования в алгоритме RSA***

Предположим, что пользователь *А* хочет передать пользователю *В* сообщение в зашифрованном виде, используя алгоритм RSA В таком случае пользователь *A* выступает в роли отправителя сообщения, а пользователь *В -* в роли получателя. Как отмечалось выше, криптосистему RSA должен сформировать получатель сообщения, то есть пользователь *В.* Рассмотрим последовательность действий пользователей *В* и *А:*

1. Пользователь *В* выбирает два произвольных больших ***простых*** числа *Р* и *Q.*
2. Пользователь *В* вычисляет значение модуля *N= P\*Q.*

3. Пользователь *В* вычисляет функцию Эйлера

*φ(N)=(P -1)(Q - 1)*

и выбирает случайным образом значение открытого ключа *КB* с учетом выполнения условий

*1<КВ ≤ φ(N,* , НОД *(КB, φ(N))*= 1.

4. Пользователь *В* вычисляет значение секретного ключа *kB,* используя расширенный алгоритм Евклида при решении сравнения



5. Пользователь *В* пересылает пользователю *А* пару чисел (.*N*, *Кв)* по незащищенному каналу.

Если пользователь *А* хочет передать пользователю *В* сообщение *М,* он выполняет следующие шаги:

6. Пользователь *А* разбивает исходный открытый текст *М* на блоки, каждый из которых может быть представлен в виде числа

М, = 0, 1, 2, ..., *N —* 1.

7. Пользователь *А* шифрует текст, представленный в виде последовательности чисел *М,* по формуле



и отправляет криптограмму

*С1, С2, С3 ….. Сi ……..*

пользователю *В.*

1. Пользователь *В* расшифровывает принятую криптограмму

*С1, С2, С3 ….. Сi ……..*

используя секретный ключ *kB,* по формуле



В результате будет получена последовательность чисел *М„* которые представляют собой исходное сообщение *М.* При практической реализации алгоритма RSA необходимо иметь возможность без существенных затрат генерировать большие простые числа, уметь оперативно вычислять значения ключей *КB* и *kB.*

**Пример: Шифрование сообщения CAB.** Для простоты вычислений будут ис­пользоваться небольшие числа. На практике применяются очень большие числа (длиной 250-300 десятичных разрядов).

*Действия пользователя В:*

1. Выбирает Р = З и Q = 11.
2. Вычисляет модуль N=P\*Q=3\*ll = 33.

3. Вычисляет значение функции Эйлера для *N* = 33:

*φ(N)* = φ(33) = *(Р* - 1) (Q - 1) = 2 х 10 = 20.

Выбирает в качестве открытого ключа *КB* произвольное число с учетом выполнения условий

*1<КВ<* 20, НОД *(КB,* 20) = 1.

Пусть *КB* = 7.

4. Вычисляет значение секретного ключа *kB,* используя расширенный алгоритм Евклида при решении сравнения

*kB = 7-1* (mod 20).

Решение дает *kB* = 3.

5. Пересылает пользователю *А* пару чисел *(N= 33, Кв = 7).*

*Действия пользователя А:*

1. Представляет шифруемое сообщение как последовательность целых чисел в диапазоне 0-32. Пусть буква А представляется как число 1, буква В – как число 2, буква С - как число 3. Тогда сообщение CAB можно представить как последовательность чисел 312, то есть М, = 3, *М2* = 1, М3 = 2.
2. Шифрует текст, представленный в виде последовательности чисел *М1 М2* и *М3,* используя ключ *КB* = 7 и *N* = 33, по формуле

С*i* =

Получает

С, = 37(mod 33) = 2187(mod 33) = 9,

*С2* = I7(mod 33) = l(mod 33) = 1,

*С3,* = 27(mod 33) = 128(mod 33) = 29.

Отправляет пользователю *В* криптограмму *С1 С2,* С3 = 9, 1, 29.

*Действия пользователя В:*

*8.* Расшифровывает принятую криптограмму С1, С2, С3, используя секретный  
ключ *kB* = 3, по формуле



Получает

М1 = 93(mod 33) = 729(mod 33) = 3,

М2 = l3(mod 33) = l(mod 33) = 1,

*М3* = 293(mod 33) = 24389(mod 33) = 2.

Таким образом, восстановлено исходное сообщение CAB: 312.

Криптоалгоритм RSA всесторонне исследован и признан стойким при достаточной длине ключей. В настоящее время длина ключа 1024 бит считается приемлемым вариантом. Некоторые авторы утверждают, что с ростом мощности процессоров криптоалгоритм RSA потеряет стойкость к атаке полного перебора. Однако увеличение мощности процессоров позволит применять более длинные ключи, что повышает стойкость RSA. Следует отметить, что алгоритм RSA можно применять как для шифрования сообщений, так и для электронной цифровой подписи.

Нетрудно видеть, что в асимметричной криптосистеме RSA количество используемых ключей связано с количеством абонентов линейной зависимостью (в системе из *N* пользователей используется 2 х *N* ключей), а не квадратической, как в симметричных системах.

Сравнивая наиболее популярных представителей асимметричного и симметричного шифрования, следует отметить, что быстродействие RSA существенно ниже быстродействия DES, а программная и аппаратная реализация криптоалгоритма RSA гораздо сложнее, чем DES. Поэтому криптосистема RSA как правило, используется при передаче сообщений небольшого объема.

2. Практика.

Составьте программное обеспечение, реализующее алгоритм RSA. Исходные данные должны передаваться через файлы: файл с открытым ключом, закрытым ключом и шифруемая информация. Для созданного программного обеспечения проведите тестирование не менее чем на 10 различных наборах данных.