**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И**

**ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

**и варианты заданий**

### К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

### ПО ДИСЦИПЛИНЕ

### «Информационная безопасность»

### Казань, 2019

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Название работы.** *Реализация в среде программирования алгоритма RSA шифрования с открытым ключом.*

**Цель.** Ознакомиться с алгоритмом шифрования с двумя ключами RSA.

**Задание на лабораторную работу:**

1. Изучить теоретический материал по данной лабораторной работе.
2. Разработать и написать в одной из сред программирования и **любом** языке программу шифрования по методу RSA. Требования к выполнению работы изложены ниже.
3. Выполнить шифрование/расшифровку конфиденциальных данных.
4. Ответить на контрольные вопросы в конце задания (устно при сдаче задания).

**Указания к выполнению лабораторной работе.**

**Программа реализация метода RSA должна уметь работать с длинными целыми числами. Она должна иметь модульный характер и содержать следующий набор процедур, которые можно запускать отдельно (с помощью оконного интерфейса):**

* **Реализацию расширенного алгоритма Евклида.**
* **Функцию быстрого возведения в степень по модулю заданного n.**
* **Функцию генерации простых чисел с проверкой простоты с помощью метода Миллера –Рабина. Простые числа, предназначенные для RSA, должны генерироваться с помощью датчика случайных чисел и иметь заданную длину (например, десять десятичных разрядов). Необходимо, чтобы каждое простое число *p*, генерируемое для RSA, удовлетворяло *следующему условию*: число *p-1* должно содержать простой множитель больший 106. Последнее требование необходимо для противодействия разложению параметра n на простые сомножители (p-1)-методом Полларда.**
* **Модуль генерации параметров RSA длины, заданной пользователем с выводом данных на экран.**

**Основная функция программы – выполнение шифрования/расшифровки произвольного текста, вводимого с клавиатуры или из текстового файла. Для шифрования текст разбивается на отдельные блоки символов, каждый из которых заменяется кодом, значение которого не превышает величину параметра n=p\*q и шифруется путем возведения в степень e по модулю n. Пример кодирования текста – в лабораторной работе 2.**

# 

# Теоретический материал

***Односторонняя (однонаправленная) функция (one way function)*** - это функция *f*, осуществляющая отображение **X**->**Y**, где **X** и **Y** - произвольные множества, и удовлетворяющая следующим условиям:

1. Для каждого x из области определения функции  легко вычислить . Понятие «легко» обычно означает, что существует алгоритм, вычисляющий функцию f(x) за полиномиальное время от длины аргумента x.
2. Задача нахождения прообраза  для произвольного y, принадлежащего области значений функции, является вычислительно сложной задачей. Последнее означает, что не существует алгоритма, вычисляющего  существенно быстрее, чем алгоритм полного перебора.

Задача разложения натурального числа N на простые множители (*факторизация* N) является задачей вычисления односторонней функции: зная сомножители p и q, нетрудно вычислить их произведение N=p • q, но обратная задача нахождения делителей p и q по известному N является сложной задачей, решение которой требует значительных вычислительных ресурсов.

На вычислительной сложности решения этой задачи построен один из самых известных асимметричных методов криптографии – метод RSA. В 1977 году, когда создатели этого метода Ривест, Шамир и Адлеман объявили о новом методе криптографии, основанном на задаче факторизации, наиболее длинные числа, разложимые на множители, имели длину 40 десятичных цифр, что соответствует, примерно, 132-битовому двоичному числу (число 40 надо домножить на ). Создатели RSA предложили широкой публике расшифровать некую фразу английского языка. Для этого предварительно требовалось факторизовать 129-значное десятичное число N (длины 428 в битовом представлении), про которое было известно, что оно представимо в виде произведения двух простых сомножителей p и q, которые имели длину 65 и 64 десятичных знака.

С тех пор был достигнут значительный прогресс в этой области. Число, предложенное создателями RSA, было разложено в 1994 году с помощью нового мощного метода факторизации – метода квадратичного решета (Quadratic Sieve Factoring), разработанного Карлом Померанцем и реализованного Аткинсом, Граффом, Ленстрой и Лейлендом. В работе участвовало около 600 добровольцев, задействовано в сети около 1700 компьютеров, которые работали в течение 7 месяцев.

Параллельно с этим методом Джоном Поллардом, известным специалистом по криптографии и теории алгоритмов, был разработан еще более быстрый метод, получивший название метода решета числового поля (Generalized Number Field Sieve - GNFS), который является наиболее быстрым методом и на сегодняшний день. Текущий рекорд, установленный немецкими исследователями, на июнь 2008 года, составляет 1000-бит. Это делает небезопасными ключи RSA длины 1024, которые являются на сегодняшний день, самыми распространенными.

Задача шифрования по методу RSA состоит из двух самостоятельных задач: первая из них – генерация параметров метода, т.е пары «открытый/закрытый ключ»;

вторая – собственно шифрование и расшифровка данных с помощью сформированных ранее ключей. Рассмотрим каждую задачу более подробно.

***Генерация пары «открытый/закрытый ключ» для метода RSA.***

1. Выбираются два простых числа p и q. Для нашего примера числа p и q должны иметь заданную длину до 1024 бит. В этом случае их произведение N=p • q будет иметь длину до 2048 бит. Для тестирования выбираются значения меньшей длины.
2. Вычисляем их произведение N=p • q.
3. Вычисляем функцию Эйлера .

*(Значение равно числу натуральных чисел, меньших N, взаимно простых с , (т.е. числу всех натуральных чисел k < N таких, что наибольший общий делитель НОД(N; k)=1).*

1. Задаем параметр e, входящий в открытый ключ RSA, равным произвольному числу, меньшему N, взаимно простому с . При реальном шифровании длина e выбирается приблизительно равной L/3, где L – длина N. Можно взять e, равным произвольному простому числу, меньшему N и отличному от p и q, проверив при этом условие того, что  не делится на е ().
2. Находим параметр d, являющийся секретным параметром метода RSA, из условия . Для вычисления d необходимо воспользоваться *расширенным алгоритмом Евклида*, который описан ниже. Расширенный алгоритм Евклида по входным натуральным числам A и B находит их наибольший общий делитель C=НОД(А,В), а также числа x и y такие, что выполнено равенство .

*Для нахождения параметра d подставим в расширенный алгоритм Евклида вместо входных чисел А и В числа  и е. Выполнив вычисления по алгоритму Евклиду мы найдем числа x и y такие, что . Применив операцию к обеим частям последнего равенства, получим . Значит, можно взять значение параметра d равным коэффициенту y из метода Евклида. Однако, коэффициент y может принимать как положительные, так и отрицательные значения, а параметр d обязательно должен положительным, поэтому в случае если y < 0, следует взять *

Генерация параметров RSA завершена. Пара (N, e) объявляется открытым ключом, а параметры  и d закрытыми параметрами (d – закрытый ключ RSA).

***Шифрование/расшифровка сообщений по методу RSA*.**

Текст, который следует зашифровать, разбивается на отдельные блоки символов L – длина N в битовом представлении). Например, если каждый символ текста имеет код длины m бит, то блок, состоящий из k символов, можно закодировать числом длины k\*m бит.

В лабораторной работе 2 описана кодировка, в которой символ кодируется двухзначным десятичным числом. Секретное слово длины, например, 12 символов кодируется числом длины 24 десятичных разряда или примерно 80 бит. В этом случае, длина n=pq должна иметь не менее 80 бит. Если n имеет меньший размер, то следует шифровать блоками меньшей длины.

***Пример.*** Кодируем буквы английского алфавита числами от 1 до 56 (заглавные и малые буквы), плюс цифры и знаки препинания. Например, слово Hello имеет код

Hello -> 08 33 40 40 43.

Для шифрования числа *с*, служащего кодом блока текста, выполняется операция возведения в степень по формуле

 (\*)

где числа N, e взяты из открытого ключа RSA.

Обратная расшифровка выполняется возведение шифра r в степень d, где d – секретный параметр RSA.

 (\*\*)

Гарантией того, что при повторном возведении в степень, будет получено исходное число, служит теорема Эйлера, которая утверждает, что для любого выполняется формула . Проверим операцию (\*\*):



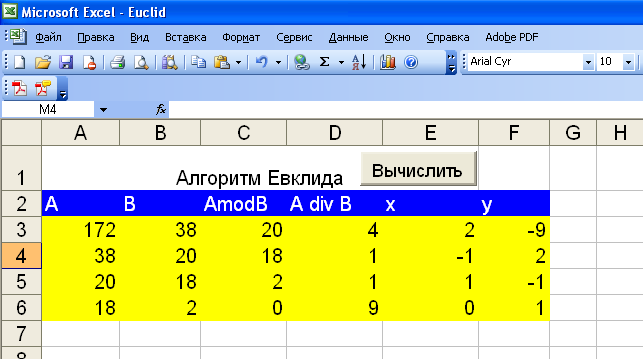
***Расширенный алгоритм Евклида.***

**Алгоритм *Евклида*** используется для нахождения по заданным целым числам A и B их наибольшего общего делителя С=НОД(А; B). ***Расширенный*** алгоритм Евклида используется также для нахождения целых чисел x, y таких, что выполняется условие . Используется во многих криптографических конструкциях, в том числе в методе RSA.

Основным равенством, используемым для вычисления НОД чисел А и В, является условие

НОД(А, В) = НОД( В, A mod B) (3)

где A mod B – остаток от целочисленного деления А на В. Применяя последовательно формулу (3), мы будем уменьшать числа А и В в этом алгоритме, пока A mod B не станет равным 0, тогда последнее значение аргумента В даст искомый НОД (А, В). Полное описание расширенного алгоритма мы объясним на примере. Пусть числа А и В равны 172 и 38 соответственно. Откроем рабочий лист Excel и разметим в первых строчках заголовки столбцов, а под заголовками А и В поместим числа 172 и 38, как указано на рисунке.



**Рис.1.** Примерный вид рабочего листа Excel для реализации расширенного алгоритма Евклида.

В столбце Amod B напишем формулу вычисления остатка от целочисленного деления А на В: =ОСТАТ(A3;B3), а в столбце A div B впишем формулу =ЦЕЛОЕ(A3/B3), обозначающую операцию нахождения целой части от деления A на В.

В следующей строке в столбах А и В поместим значения 38 и 20, взятые из двух ячеек, находящихся выше и правее (применение формулы (3)). Значения ячеек под заголовками Amod B и A div B надо вычислить как в предыдущей строке. В Excel для этого достаточно просто скопировать и вставить вышележащие клетки на строку ниже.

Повторяем эти операция несколько раз, пока не получим в столбце Amod B значение 0. Значение В в этой строке будет равно НОД(А,В) (в нашем примере он равен 2).

Далее заполняем столбцы x и y в обратном порядке снизу вверх. Поместим в столбцы x и y в последней полученной строке значения 0 и 1. Далее в каждой следующей (вышележащей) строке i поместим значения xi и yi, вычисленные по формулам:



где  обозначает значение из столбца A div B, взятое из той же строки, где вычисляются значения x и y. Используя эти формулы, заполним столбцы x и y. В верхней строке получим значения x и y, которые нам нужны.

***Алгоритм возведения в степень по модулю натурального числа.***

Для выполнения шифрования по методу RSA приходится выполнять возведение в большую степень различных чисел, а результат приводить по модулю числа N, являющегося параметром метода и имеющего длину 512 и более бит. Уже для небольших ***a*** и ***e*** вычислить значение

 (1)

выполняя сначала возведение в степень, а потом вычисляя остаток от деления  на N, становится невозможным. Между тем, если применить алгоритм, описанный в этом разделе, можно вычислять выражения (1), для достаточно больших чисел a, e, N, оставаясь в рамках обычных операций с целыми числами, заложенных в компьютере.

Алгоритм быстрого возведения в степень основывается на идее замены прямого вычисления возведения в степень последовательными операциями умножения на a и возведения в квадрат. Для этого представим степень e число в двоичном исчислении

 (2)

где любое ti для  принадлежит , . Зная вектор разрядов , можно вычислить число *e*, применяя последовательные вычисления:

,  (3)

Например, если *e* = 13, то в двоичном представлении *e* = 11012 , и 13 можно представить как результат вычисления  , .

Последнее число и есть *e*.

Используя формулы (3), можно процедуру возведения в степень по модулю натурального числа N, записать в виде последовательности итераций:



где . Множитель  в зависимости от  принимает либо значение *a,* если , либо 1, если . Результат вычислений можно свести в таблицу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | ... |  |
|  |  |  | ... |  |

***Пример.*** Вычислить .

***Решение.*** Переведем степень e=13 в двоичный вид. Для этого заполним следующую таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| e div 2 | 13 | 6 | 3 | 1 |
| e mod 2 | 1 | 0 | 1 | 1 |

***Таблица 1.*** Перевод десятичного числа e к двоичному представлению.

Искомое двоичное разложение числа *e* будет во второй строке таблицы , записанное в обратном порядке справа налево.

Далее, составим таблицу вычисления с, заполняя следующую таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| е | 1 | 1 | 0 | 1 |
| с | 5 | 11 | 7 | 17 |

***Таблица 2.*** Возведение а=5 в степень e=13 по модулю 19.

В первой строке запишем цифры двоичное разложения числа 13. В первую ячейку второй строки поместим основание *а*=5. Далее каждое следующее значение *с* будем вычислять по формуле:



Например,







**Генерация простых чисел**

Для определения параметров RSA необходимо генерировать простые числа заданной длины. Согласно теореме Чебышева о распределении простых чисел число простых чисел на интервале [1; N ] асимптотически стремится к . Для небольших N значение функции π(N) может сильно отличаться от указанной величины. Например, π(106) равно примерно 78,5 тысяч. Для генерации простых чисел можно выбрать произвольное нечетное число T и проверить его на простоту с помощью одного из тестов, описанных ниже. Если T окажется составным, то надо заменить его на T+2 и проверить снова затем проверить T+4 и т.д. В среднем, через (ln\_T) шагов простое число будет найдено.

**1. Перебор делителей числа Т.** Если число Т – составное, то найдется число k, меньшее , которое делит Т. Поэтому простейшим тестом на простоту является проверка делителей числа Т от 2 до . Приведем программу на С:

int Check\_prime( long T){

**int**  k = int(Sqr(double(T));

For (i = 2; i< k;i++)

If (T % i==0)

return i; // T имеет делитель i

return 0; // Т – простое число

}

**2. Тест Ферма.** Согласно теореме Ферма, если число Т – простое, то для любого . На обращении этой теоремы основан следующий вероятностный тест:

Если для произвольных различных k чисел *a*, меньших T, выполняется условие , то с вероятностью, не меньшей, чем , число *a* является простым.

К сожалению, полностью обратить теорему Ферма нельзя, т.к. существуют так называемые числа Кармайкла, для которых условие Ферма  выполнено для всех *a*, меньших T, но числа являются составными. Однако, поскольку числа Кармайкла встречаются довольно редко, то в наших условиях можно ими пренебречь.

***3. Тест Миллера Рабина.*** Пусть R – произвольное натуральное число. Представим R–1 в виде 2s∙t, где t – нечетно. Пусть ***a***< R–1 – натуральное число. Будем говорить, что число ***a*** отвергает простоту R, если выполнено одно из двух условий:

а) R делится на ***a***,

б)  и для всех целых k,  .

Если найдется число ***а***, отвергающее R, то R является детерминировано составным. Иначе, число ***a*** подтверждает гипотезу о простоте числа R (т.е усиливает вероятность того, что R является простым). Поэтому тест Миллера-Рабина является вероятностным.

Тест Миллера состоит из отдельных проверок для различных a < R–1 и выполняется следующим образом:

1. Выполняем разложение R–1 = 2s∙t, где t – нечетное число.
2. Выбираем случайное число **a**, 1 < a < R-1. Проверим, что R не делится на а нацело. Если делится, то R – составное. Иначе, продолжим тест.
3. Вычисляем . Если b равно 1, то R – вероятно простое число. Продолжаем тест со следующим ***a***. Иначе, перейдем к следующему пункту.
4. Последовательно выполняем итерацию  до тех пор, пока b не станет равным R–1, либо число итераций не превысит s-1, где s взято из пункта 1.

Если раньше выполнится b = R–1, то число R – вероятно простое, иначе R – составное.

1. Повторим процедуру с новым ***а***.

После k циклов вероятность того, что R – составное, меньше 4-k, т.е. убывает очень быстро.

**Разложение чисел на множители методом ρ-эвристики Полларда.**

Этот метод факторизации натурального числа N был разработан известным криптографом Джоном Поллардом и является самым быстрым среди простых методов факторизации. Его идея состоит в том, что порождается случайная последовательность чисел {xi} (например, взяв x0=2, и продолжить вычисление по формуле ). Далее, вычисляются всевозможные разности . Для каждой такой пары (xi, xj) находятся с помощью алгоритма Евклида наибольший общий делитель НОД(N, |xi - xj|)). Если будет найдена пара (xi, xj) со свойством НОД(N, |xi - xj|))>1, то этот НОД и даст искомый делитель числа N.

**Замечание.** Вычисление всевозможных разностей |xi - xj| требует хранения в памяти компьютера всех промежуточных значений xi , что отнимает много памяти, поэтому используется модификация Флойда, когда из всех xi , удовлетворяющих условию  вычитается только одно значение xj  для j=2k (см.пример ниже). В интерпретации Флойда вводится новая переменная y, текущее значение которой равно предыдущему xj, вычисленному на шаге j=2k, и на каждом шаге i вычисляется разность d=|xi - y| и НОД(N, d) до тех пор, пока этот НОД не окажется больше 1, тогда он станет равным искомому делителю N.

***Обоснование метода Полларда.*** Пусть p –делитель N. Среди членов последовательности {xi} рано или поздно попадутся числа xi и xj такие, что =  (это равенство записывается более кратко ). Это условие означает, что  для некоторого целого числа k. Т.к. p – делитель N, то для выбранной пары (xi, xj) НОД(N; |xi - xj|)= p.

**Оценка сходимости метода Полларда.** Т.к. меньший делитель числа N меньше , то элементы последовательности  меньше . По парадоксу близнецов (см. [1]) среди первых  членов последовательности  с вероятностью, большем чем 0,5, попадутся два одинаковых члена, т.е. найдутся i, j, удовлетворяющие условию . Поэтому, с вероятностью, большей 0,5, мы найдем искомый делитель N, просмотрев не более , членов последовательности.

Рассмотрим пример. Пусть N=1387. Зададим рекуррентную последовательность чисел {xi} следующим соотношением: =2, . Значения строки xj определим равными ближайшему слева xi, у которого i является степенью 2 (такие xi выделены красным). В следующей строке поместим их разность, а в последней строке – НОД(N; |xi - xj|), вычисленный с помощью алгоритма Евклида.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итер.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **xi** | **2** | **3** | **8** | **63** | **1194** | **1186** | **177** | **814** | **996** | **310** | **396** |
| **yi** | **1** | **2** | **3** | **3** | **63** | **63** | **63** | **63** | **814** | **814** | **814** |
| **|xi-yi|** | **1** | **1** | **5** | **60** | **11131** | **1123** | **114** | **751** | **182** | **504** | **418** |
| **НОД** | **1** | **1** | **1** | **1** | **1** | **1** | **19** | **1** | **1** | **1** | **19** |

Из таблицы видно, что уже на 7-м шаге был найден делитель N, равный 19.

**Литература:**

1. Ш.Т. Ишмухаметов. Методы факторизации натуральных чисел: учебное пособие, Казань, КФУ, 2011. Электронная версия http://old.kpfu.ru/f9/bibl/Monograph\_ishm.pdf
2. Ишмухаметов Ш.Т. Технологии защиты информации в сети. Курс лекций.

<http://ksu.ru/f9/bin_files/metod_tzis!113.doc>

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что вычисляет классический алгоритм Евклида?
2. Сколько строчек вычислений необходимо произвести в алгоритме Евклида, если число B состоит из 3 десятичных цифр?
3. Как производится заполнение столбцов x и y в расширенном алгоритме Евклида?
4. Какая алгоритмически сложная задача лежит в основе метода RSA?
5. Что такое простое число? Какие методы проверки простоты числа вы знаете?
6. Является ли тест Миллера-Рабина детерминированным тестом проверки простоты, или он может давать ошибки?
7. На что влияет число итераций в тесте Миллера-Рабина? Как оно выбирается?
8. Как сгенерировать простое число *p* длины 10 разрядов, такое что p-1 имеет простой делитель длины >106 ?
9. Как генерируется параметры метода RSA?
10. Какие параметры в RSA вычисляются с помощью алгоритма Евклида?
11. Как производится процедура шифрования/расшифровки в методе RSA?
12. Какой длины должны быть простые числа p и q в методе RSA, чтобы обеспечить необходимый уровень надежности?
13. Каким образом, зная значение функции Эйлера и открытый ключ е, можно рассчитать закрытый параметр d?
14. Дайте определение односторонней функции.
15. Сколько итераций потребуется приблизительно сделать в методе Полларда, если N ≈108?
16. Сколько памяти требует для разложения n-битового числа оригинальный метод Полларда? Объяснить модификацию Флойда метода Полларда.
17. Сколько времени потребуется современному персональному компьютеру, чтобы разложить на множители десятичное число, занимающее на экране одну строку (80 позиций)?

**Лабораторная работа №2**

***Взлом RSA с помощью факторизации ключа N.***

Во второй работе надо разработать программу, выполняющую поиск секретных параметров метода RSA, путем разложения известного открытого ключа N и параметра e. Для разложения числа на множители надо применить один из следующих методов (pho-метод Полларда, (p-1)-метод Полларда, метод эллиптических кривых или другой). Необходимо найти такие параметры, как общее время работы программы, число итераций. Кодирование и раскодирование текста выполняется, как описано ниже. Вариант заданий взять согласно своему номеру в списке группы (потока).

**Кодирование слов.**

<https://ru.wikipedia.org/wiki/Windows-1251>

В кодировке Win1251 русский алфавит занимает место с позиции от А=410 16=1040 10 (“А” заглавное) до я = 44F16 = 110310 ( “я” строчное). Для упрощения кодирования из кодировки win 1251 было вычтено число 1024=400 16, поэтому русский алфавит занял позиции с 16 по 79:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ы | Ь | Э | Ю | Я |
| 43 | 44 | 45 | 46 | 47 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | б | в | г | д | е | ж | з | и | й | к | л | м | н |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| о | п | р | с | т | у | ф | х | ц | ч | ш | щ | ъ |
| 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ы | ь | э | ю | я |
| 75 | 76 | 77 | 78 | 79 |

Далее, слово было закодировано числами по основанию 100. Например, слово «Год» кодируется числом 19\*1002+62\*100+52=196252. Значит, для раскодирования секретного слова x необходимо вычислить секретное число, равное



где x=SW и N заданы в Вашем варианте, параметр d является секретным ключом, который надо найти, факторизуя N в произведение двух сомножителей вычисляя и найти d из уравнения



Когда число y будет найдено, его надо разбить на пары цифр и восстановить слово по кодовой таблице, приведенной выше.

**Варианты заданий**

Вариант 1

n=344572667627327574872986520507, e=303498613397661458186613409505, SW=42393687358080034300726554172

Вариант 2

n=374446690081577726518750680131, e=227308426243091608838383020209, SW=173544927178205874489521719542

Вариант 3

n=2015638500984252513380255580199, e=1118845478227818565358376981641, SW=1968403772706991943140704820969

Вариант 4

n=2803016870163537170844910120589, e=321029592955180891096364204933, SW=14592629850394393337941666131

Вариант 5

n=801354636919526323408669959133, e=5357133285262456827514698587, SW=630839280032278632036025925797

Вариант 6

n=875078015194165779807004996891, e=3317761452838260724386723229, SW=736612590537995601962451694817

Вариант 7

n=2945551046056896373232621994521, e=1578240369330316947030869051291, SW=2268046305917896117885320940385

Вариант 8

n=342009136322315980094889308153, e=208882210355060386981711608527, SW=323071158829636363902958789743

Вариант 9

n=1204053537389703721541732260277, e=861912432659932141657251942349, SW=536925649708611072738514452088

Вариант 10

n=54532674247289454314563008683, e=11477646364080609704857416529, SW=43355907199058798176130579652

Вариант 11

n=4442502401588849632243368606511, e=1190181981555380137698609038857, SW=2194663810499391809566551485933

Вариант 12

n=88008953914041855699205104599, e=47229555628553041136830405571, SW=47407773157716183186429902491

Вариант 13

n=86892104384358621580698011597, e=26398962489114266337226913921, SW=62873526787664405062856709109

Вариант 14

n=279987870740665381618843285073, e=118086993812917654371624726349, SW=162836163952303018366729425385

Вариант 15

n=1441006610206833289007830556447, e=1102261365739821522530333068003, SW=694624368581721113652509149602

Вариант 16

n=310006596248373369847685371499, e=131417437022740390789727023183, SW=73207463908856286134350964021

Вариант 17

n=412950519942098514204399552799, e=34343127213212244935444124191, SW=221300658194014757382986496783

Вариант 18

n=1303470844272911466854652379811, e=621046881056292008081324361643, SW=1157594636053203376021925344632

Вариант 19

n=365986541659725380455392101587, e=345463653468664133151171563183, SW=227618660835758378742072466215

Вариант 20

n=3261434502421303279323564402143, e=2939271343282951673253104740745, SW=1729120759169116226435526469188

Вариант 21

n=880256638333804972652092375741, e=679965420900654922416860968339, SW=797932992062154509068500137757

Вариант 22

n=5087308846537040504749735052677, e=635964111226216674951775514511, SW=3009767917863379873938622968631

Вариант 23

n=59824481984506504151410133849, e=9396366867643299157738812133, SW=38815413112101541429153780760

Вариант 24

n=345643290359673252210751286509, e=42230327764300114870970880547, SW=100986129396953142858650179714

Вариант 25

n=240719707794161389931571470417, e=1594459547134895900616767861, SW=214377409021092070327230236385

Вариант 26

n=2382108426543348401296729081237, e=1154894534459475385454702210381, SW=1273370590783719809245106145607

Вариант 27

n=329814218084908451102237952889, e=66076832588057025959339642363, SW=114675515992271270646220477408

Вариант 28

n=1375719758603974834448145016229, e=786912904386116647910035357501, SW=438444967054424668764166860047

Вариант 29

n=1255621015589767357963616484373, e=1229439078303093414729367665997, SW=6707128251645275107007709510

Вариант 30

n=4503580535302153411490266610041, e=1189393447683752946756622398941, SW=2570503345929776816031645086354

Вариант 31

n=1627040944314653910634318496023, e=180426666459182597612758185881, SW=390704174174367535084038714741

Вариант 32

n=3576943277408979005059757821217, e=3482204036556926093749575408709, SW=1123015711801071636107843562874

Вариант 33

n=3260730192713051360762140398937, e=1208481207980698001137312191587, SW=1355838234634039892969275383829

Вариант 34

n=2982927654040643617168221295807, e=2873746014960179413091709562637, SW=476789675161565579480825665424

Вариант 35

n=274367968163392106602848256127, e=154899187893923739338714676739, SW=64572423133411071142293616868

Вариант 36

n=1759704387935284899984083501633, e=1668350266552664908136290878215, SW=101826364771542867589783519268

Вариант 37

n=50521984138040381699131985921, e=1662206736409154854019667059, SW=9132458113454720873580260016

Вариант 38

n=2887972237802982718212271315873, e=157998649488421547233695490521, SW=1996567031029429901473983017849

Вариант 39

n=2787714501387687333564375076367, e=299520927037692620509634283853, SW=2081392748597597074193254643883

Вариант 40

n=73251552185971008095619995381, e=11990006860577370723253958291, SW=13067984531505580476221665929

-------------------------------------------------------

Вариант 41

n=2143329688117157614570671479917, e=280157286011511271533989422951, SW=897283893756127578322292118958

-------------------------------------------------------

Вариант 42

n=3485198031263125658050790686801, e=3352025175036927419876530203125, SW=2973056604120304522969999696491

Вариант 43

n=770558893884724598529533674937, e=722328504495843807022833112153, SW=335511379427534311656710798968

Вариант 44

n=1577271624417732056618338337651, e=1291637051908295587224621302305, SW=13564020148294776742944381452

Вариант 45

n=502535245393651877650828332721, e=182141162869635425214823416737, SW=87988840834435229649551090334

Вариант 46

n=309990562161532629109378549703, e=296584741425508269897752766319, SW=74916456162242222413936537236

Вариант 47

n=243513455644316518701642102143, e=239771145220867991202203659549, SW=11900361514720697833561840495

Вариант 48

n=1201742566949757754899862890233, e=641647860430765751458392375359, SW=341217881604209907874675821433

Вариант 49

n=507209718350630004049694027123, e=184487185821747957740702302621, SW=63422596028358682971997977073

Вариант 50

n=469066560453237099419615679323, e=420335640180846896361633610321, SW=318792603074051503529131980007

Вариант 51

n=3603193717203092917877506797763, e=2330934552496643682346076072881, SW=2462962878231895456628444926931

Вариант 52

n=405421750182999546086416639213, e=352664141381672526074673384969, SW=190482502633602522026716406210

Вариант 53

n=537403628975761650764156954993, e=517214694290847882761450046499, SW=26995737547564296500405026625

Вариант 54

n=309945939017081359552445591591, e=256216931276866843653727191713, SW=125965184859491501024252798356

Вариант 55

n=2117319396463620915099437881223, e=1105376787388722496862806077385, SW=1190288137632174072573528855141

Вариант 56

n=1122198810262474384600312231187, e=63748139006231344946621145973, SW=292017007173513613151239211267

Вариант 57

n=1644674509297606959628451803057, e=997586802469912185830957414599, SW=550843275631403134960714469840

Вариант 58

n=75404363540366760439727901499, e=33451835917331251417201445711, SW=32517816964116314719497361317

Вариант 59

n=73807430560852047360352292987, e=38707316417589116492583653317, SW=19211261404195575283252447332

Вариант 60

n=1764776284484729897892738967891, e=619104589685425039468374670087, SW=1319592008578141450291094904313

Вариант 61

n=2425742060723730952266324124559, e=324709972887192578587817677057, SW=919206802719720879838473172161

Вариант 62

n=235776045139618218610341863683, e=52020494195584045512061375287, SW=153276312000303685717028522325

Вариант 63

n=72399694682825837946927975157, e=66583587191012908259630071497, SW=65408072063058019072253577314

Вариант 64

n=82226843075939372862059256271, e=39574931997271565078938293107, SW=72299210235764206342642689453

Вариант 65

n=576465851348833980669060214423, e=263373400824198297846758495363, SW=222695874265921322617697906073

Вариант 66

n=1309320295521959979486637519751, e=710818076151030423063628477991, SW=1151643322579967235272288665179

Вариант 67

n=388073116243299719049697041059, e=93774658102057243875664542319, SW=31310614758708575891098216049

Вариант 68

n=106171883841886625367977923841, e=47953521058903799911249297975, SW=7236098763031435159742573134

Вариант 69

n=75332357154462380976079586039, e=24639182129584606917471570503, SW=26752566085475942776180898092

Вариант 70

n=73764210312306688035707750563, e=23526777810187440967780547753, SW=9502486013590286668593653859

Вариант 71

n=406701666106963162281904899119, e=57927194748969572194379255821, SW=320154759777551452843644098149

Вариант 72

n=224251103908338188528711756183, e=171879645203279162794009958633, SW=67407413582329338256717213300

Вариант 73

n=3991181910811876034905559121593, e=2413311687709450222212161769949, SW=909935708517973157565471317180

Вариант 74

n=73928320426901978411951089411, e=2402688671386997131627458187, SW=20011660086946308434694374791

Вариант 75

n=69736417863105965026610220973, e=34275792357732740887562394731, SW=51083116293135572495621698283

**Лабораторная работа №3.**

**Название работы.** *Разработка клиент-серверного приложения.*

**Цель работы:** Изучить современные средства создания клиент-серверных приложений в системе VS 2010 или более поздней (не обязательно!). Научиться практической работе по организации и решению задач информационной безопасности в сети.

**Задание на лабораторную работу**. 1. Разработать, используя среду программирования клиент-серверное приложение для двустороннего обмена информацией между компьютерами в сети. Клиентское и серверные приложения должны выполнять следующие функции:

1. Регистрацию новых пользователей на сервере в БД MySQL или подобной.
2. Выполнять аутентификацию пользователей по схеме «слово-вызов» с безопасной передачей паролей.
3. Выработать секретный ключ по протоколу Диффи-Хелмана для шифрования/расшифрования данных по методу RC4 (потоковый метод).
4. Выполнять проверку целостности текстов с помощью цифровой подписи, вычисляемой с помощью метода RSA (использовать лабораторную работу 1).

**Требования к выполнению задания.** Клиентское приложение должно содержать форму, на которой содержатся поля для ввода IP-адреса компьютера – сервера, поле для ввода информации, передаваемое на сервер и поле для получения информации, возвращаемой с сервера.

Приложение должен содержать кнопки ***Старт/Стоп*** для запуска и остановки сервера, поле для вывода информации, передаваемой с сервера, и поля для вывода информации, передаваемой клиентами.

Приложение также должно содержать генератор ключей для протокола Диффи-Хелмана и вычисления секретного ключа.

При сдаче необходимо установить клиентскую часть на один компьютер, а серверную часть приложения на другой компьютер, и продемонстрировать диалог обмена данными.

**Программно-аппаратные средства.** Компьютерная лаборатория, состоящая из компьютеров, соединенных в локальную сеть, среда программирования типа VS 2010 или более поздняя.

**Задание на лабораторную работу**

1. Изучить теоретический материал по данной лабораторной работе.
2. Ознакомиться с указаниями по программированию на языке Pascal в среде Delphi или другом языке программирования.
3. Разработать программный комплекс, представляющий собой клиент-серверное приложение, решающее следующие задачи:

* удаленную аутентификацию пользователей по алгоритму «Вызов-Ответ» с использованием хеш-функции MD5 или аналогичной.
* генерацию общего секретного ключа по алгоритму Диффи-Хелмана.
* шифрование данных по алгоритму RSA.
* выработку сервером электронной цифровой подписи и передача ее секретным образом пользователю.

1. Выполнить пробное шифрование/расшифровку данных, передаваемых по сети в рамках компьютерного класса, выполнить аутентификацию пользователей на сервере с помощью открытого ключа ЭЦП, хранимого на сервере.
2. Вставить в отчет полученные данные, описать методику выполнения задания.
3. Ответить на контрольные вопросы в конце задания.

# Теоретический материал.

Рассмотрим в качестве примера процедуру создания приложений для обмена сообщениями в сети по протоколам TCP/IP в среде Delphi. Для разработки визуального приложения в среде Visual Studio или другой надо подобрать соответствующие компоненты.

**Разработка ТСР-сервера в Delphi.**

1. Нанесем на форму Delphi компоненту TidServer с вкладки IndyServer.
2. В его свойстве Bindings укажем IP-адрес данного компьютера и номер порта, на котором сервер будет ожидать вызова от клиента (номер порта – произвольное число от 1 до 65767, но желательно использовать номера выше 1024, т.к. порты с меньшими номерами зарезервированы для стандартных служб),
3. В свойстве MaxConnections укажите 5 (максимальное число соединений к серверу), в свойство Default Port запишите значение порта по умолчанию, а в свойство Active запишите true.
4. Добавьте на форму элемент типа TMemo для вывода в него сообщений, полученных от клиента,
5. При вызове клиента вырабатывается событие OnExecute элемента IdServer1. Для его обработка откройте вкладку Events Инспектора объектов и щелкните дважды в поле процедуры OnExecute.
6. В открывшей процедуре введите следующий код:

Procedure TForm1.IdTCTServer1Execute(Athread:TidPeerConnection);

Begin

with Athread.Connection do

begin

Memo1.Lines.Add(CurrentReadBuffer);

Writeln(‘Сообщение получено’);

Disconnect;

end;

End;

**Приложение TCT-клиент в Delphi.**

1. Нанесите на форму элемент TidTCPClient с панели IndyClient, два элемента типа Tedit для ввода сообщений серверу и получения ответа, и кнопку TButton.
2. В свойстве Host элемента TidTCPClient укажите IP-адрес сервера, а в свойстве Port задайте номер порта (тот же, что у сервера).
3. Щелкните дважды по элементу Button1 и в появившемся окне введите следующий код:

Procedure TForm1.Button1click(Sender: TObject);

Begin

IdTCPClient1.Connect;

IdTCPClient1.Writeln(Edit1.Text);

Edit2.Text:= IdTCPClient1.ReadLn;

IdTCPClient1.Disconnect;

End;

1. Добавьте код функции MD5, приложенный ниже.
2. Запустите оба приложения и протестируйте полученную программу.

# Теоретический материал по алгоритмам аутентификации.

Одной из наиболее важных служб безопасности является *аутентификация*. Аутентификация – это подтверждение пользователем информационных услуг своего идентификатора. Аутентификация выполняется с помощью разных методов, из которых простейшим является предъявления пользователем серверу секретного слова – пароля, известного только пользователю и серверу.

**Хеш-функции**

Хеш-функции играют в информационной защите важную роль, создавая для электронного документа его «моментальный снимок» и тем самым защищая документ от дальнейшей модификации или подмены.

В широком смысле функцией хеширования называется функция H, удовлетворяющая следующим основным свойствам:

1. *Хеш-функция* Н может применяться к блоку данных любой длины.
2. *Хеш-функция* Н создает выход фиксированной длины (равно, например, 128 бит для классической функции хеширования MD5, и 160 бит для функции SHA1).
3. Н (М) вычисляется относительно быстро (за полиномиальное время от длины сообщения М).
4. Для любого данного значения *хеш-кода*  h вычислительно невозможно найти M такое, что Н (M) = h.
5. Для любого данного х вычислительно невозможно найти y x, что H(y) = H(x).



1. Вычислительно невозможно найти произвольную пару (х, y) такую, что H(y) = H (x).

Термин *вычислительно невозможно* означает здесь, что в настоящее время решение этой задачи либо требует слишком большого интервала времени (например, более сотни лет), либо использования слишком больших вычислительных ресурсов, чтобы решение задачи имело смысл.

Первые три свойства требуют, чтобы *хеш-функция* создавала *хеш-код* для любого сообщения. Четвертое свойство определяет требование односторонности *хеш-функции*: легко создать *хеш-код* по данному сообщению, но невозможно восстановить сообщение по данному *хеш-коду*.

**Схемы аутентификации.**

Поскольку при передаче данных по сети никто не застрахован от возможности чтения данных на промежуточных узлах, то передача пароля по сети в открытом виде является опасным. Поэтому для надежной аутентификации и сохранения пароля от взлома используются разные схемы сетевой аутентификации. Здесь мы рассмотрим следующие три схемы:

**Схема аутентификации на основе «вызова-ответа» и хеш-функции.**

В этой схеме пользователь зарегистрировавшись на сервере, получает секретный ключ P, который сохраняется также на сервере. При выходе на связь пользователь посылает сначала свой идентификатор. Получив индентификатор, сервер проверяет наличии такого пользователя по своей базе данных и затем возвращает пользователю случайное большое число N (обычно длины 16 байт), называемое словом–вызовом. Пользователь, получив это число, формирует пару <N, P, ТS>, где P обозначает пароль пользователя, ТS – текущий момент времени (Time Stamp), подвергает ее хеш-преобразованию и отправляет полученное значение h=h(<N,P>) серверу. Сервер, получив свертку h, извлекает из базы данных пароль пользователя, выполняет то же преобразование h(<N,P) и сравнивает два полученных значения h. Если они совпали, то процедура аутентификация считается успешной.

**Схема аутентификации на основе электронно-цифровой подписи (ЭЦП).**

В этой схеме пользователь зарегистрировавшись на сервере, получает пару открытый/секретный ключ P. Открытый ключ сохраняется также на сервере. При выходе на связь пользователь формирует набор < Id, M, Тs>, где Id обозначает идентификатор пользователя, M – сообщение пользователя, Тs – метка времени (Time Stamp), подвергает его хеш-преобразованию h=h(< Id, M, Тs>) и шифрует закрытым ключом EncKз(h). Полученный код называется электронно-цифровой подписью и служит для подтверждения неизменности сообщения и проверки авторства послания. Электронно-цифровая подпись EncKo(h(< Id, M, Тs>)) прикладывается с исходному сообщению < Id, M, Тs> и отправляется на сервер.

Сервер, получив пакет, расшифровывает ЭЦП, извлекая свертку h(< Id, M, Тs>), параллельно вычисляет такую же свертку h= h(< Id, M, Тs>), используя те же исходные данные и хеш функцию, и сравнивает два полученных значения h. Если они совпали, то процедура аутентификация считается успешной. Временная метка Ts выполняет роль сеансового ключа для предотвращения атак воспроизведения.

Использование хеш-функции в этом методе не является обязательным и служит лишь для уменьшения объема вычислений для шифрования и сокращения сетевого трафика.

Электронно-цифровая подпись может быть сформирована на основе различных методов двухключевой криптографии, например, RSA, Эль-Гамаля, эллиптических кривых.

**Схема аутентификации на основе сертификата.**

Данная схема предполагает наличие третьей стороны, называемой УЦ – удостоверяющим центром или ЦС – центром сертификации, которая выдает удостоверения (сертификаты) всем участникам сетевого домена, входящего в зону действия данного ЦС. При регистрации нового пользователя или сервера в домене ЦС выдает новому участнику сертификат, состоящий из **открытой части**, содержащий такие данные как:

1. *Идентификатор владельца сертификата,*
2. *Адрес владельца сертификата,*
3. *Открытый ключ владельца сертификата,*
4. *Категория владельца сертификата (например, пользователь с ограниченными полномочиями или администратор проекта).*
5. *Наименование ЦС и его адрес,*
6. *Алгоритмы, используемые для генерации ключей и формирования ЭЦП, и их версии,*

и **закрытой части**, содержащий ту же информацию, закрепленной электронно-цифровой подписью ЦС (т.е. подвергнутого хеш- преобразованию и последующему шифрованию с помощью закрытого ключа ЦС).

Обе части выдаются соискателю в электронном виде в виде одного файла. Закрытая часть служит для того, чтобы нельзя было подделать сертификат.

Кроме того, соискатель получает отдельно закрытый ключ, соответствующий открытому ключу, находящемуся в сертификате, который соискатель обязуется хранить в секрете.

Открытая часть сертификата может быть также выдана в виде бумажного документа, подтвержденного печатью ЦС.

При обмене сообщения каждый участник сопровождает свое послание меткой времени, электронно-цифровой подписью, сформированной на основе своего закрытого ключа, и сертификатом, выданным ему ЦС. Сертификат здесь служит для удостоверения ЭЦП отправителя: получатель подписывает своим закрытым ключом послания, а получатель расшифровывает ЭЦП, используя открытый ключ отправителя, извлеченный из сертификата. Подлинность сертификата подтверждается электронно-цифровой подписью ЦС, которая может быть проверена с помощью расшифровки закрытой части сертификата с использованием открытого ключа ЦС, который является общедоступным.

**Программирование хеш-функций в Delphi**

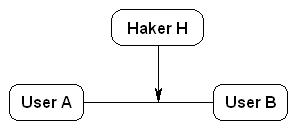
Система **Delphi** обращается к встроенным средствам операционной системы Windows для программирования различных функций хеширования, методов шифрования с использованием классической и двухключевой криптографии. Большинство этих средств содержится в библиотеках advapi32.dll и crypt32.dll, которые должны быть подключены к проекту Delphi. Для этого в проект приложения надо добавить модуль Wcrypt2.pas. О том, как воспользоваться этим модулем в проекте Delphi можно прочитать в статье Ю.Спектора

<http://www.delphikingdom.com/asp/viewitem.asp?catalogID=1271>.

Если нет необходимости использовать все возможности этих библиотек, то можно воспользоваться готовой программой для хеш-функции MD5, взятой из сети Интернет.

**Алгоритм Диффи-Хелмана выработки общего секретного ключа.**

Алгоритм Диффи-Хелмана (1976 год) (algorithm DH) служит для выработки общего секретного ключа двумя пользователями, находящимя в условиях, когда все пересылаемые ими данные, доступны третьему лицу



Алгоритм DH использует функцию дискретного возведения в степень и похож на метод Эль-Гамаля. Сначала генерируются два больших простых числа **n** и **q**. Эти два числа не являются секретными. Далее один из партнеров **A** генерирует случайное число **x<n** и посылает другому участнику будущих обменов **B** значение

**M = qx mod n**

Партнер **B** генерирует случайное число **у<n** и посылает A вычисленное значение

**K = qy mod n**

Партнер **A,** получив **K,** вычисляет **Cx = Kx mod n,** а партнер **В** вычисляет **Cy = My mod n.** Алгоритм гарантирует, что числа **Cy** и **Cx** равны и могут быть использованы в качестве секретного ключа для шифрования.

Алгоритм Диффи-Хелмана, обеспечивая конфиденциальность передачи ключа, не может гарантировать того, что он прислан именно тем партнером, который предполагается. Для решения этой проблемы можно использовать технику электронной подписи. Например, сообщения M и K, которыми обмениваются пользователи, подписываются их цифровыми подписями, что исключает подлог.

**ВАРИАНТЫ ЗАДАНИЙ**

Задание является одинаковым для всех.

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое аутентификация? Перечислите основные методы аутентификации.
2. Что такое хеш-преобразование? Перечислите основные свойства хеш-функций.
3. В чем заключается аутентификация на основе слова-вызова?
4. Что такое электронно-цифровая подпись? Как она формируется?
5. Как выполняется проверка послания, подписанного ЭЦП?
6. Что такое сертификация X.509? Каковые преимущества имеет аутентификация на основе сертификатов по сравнению с другими видами сертификации?
7. Что входит в состав сертификата ?
8. Какие основные фунции выполняется Центр Сертификации X.509?
9. Сколько различных ключей используется в процедуре аутентификация на основе сертификатов, и каким образом распространяются эти ключи?
10. Каким образом осуществляется проверка подлинности сертификата?

**ЛИТЕРАТУРА:**

1. С.Г.Баричев, В.В.Гончаров, Р.Е.Серов. Основы современной криптографии – Москва, Горячая линия – Телеком, 2001
2. А.В.Беляев. "Методы и средства защиты информации" (курс лекций). <http://www.citforum.ru/internet/infsecure/index.shtml>
3. А. А. Болотов, С. Б. Гашков, А. Б. Фролов, А. А. Часовских, «Алгоритмические основы эллиптической криптографии». Учебное пособие. М.: Изд-во МЭИ. 2000 г., 100 с.
4. А.Володин. «Кто заверит ЭЦП», журнал «Банковские системы», ноябрь 2000, <http://www.bizcom.ru/system/2000-11/04.html>
5. Т.Илонен. Введение в криптографию (Ylonen Tatu. Introduction to Cryptography), <http://www.ssl.stu.neva.ru/psw/crypto/intro.html>
6. Ш.Т. Ишмухаметов. Технологии защиты информации в сети, Казань, 2008, 91 с. http://depositfiles.com/files/e9zxcqos9
7. Н.Коблиц. Теория чисел и криптография, М.:, ТВР, 2001 <http://gabro.ge/biblio/0708/0081/file/Cryptography/Koblic_-_Teoriya_Chisel_i_Cryptografiya.rar>
8. О.Р. Лапонина. Криптографические основы безопасности, курс Интернет-университета, <http://www.intuit.ru/department/security/networksec>
9. Р.Лидл, Г.Нидеррайтер. Конечные поля, в 2 т., пер.с англ., М.: Мир, 1998, 438 с.
10. А.А. Молдовян, Н.А. Молдовян, Б.Я. Советов. Криптография. М., Лань, 2001
11. А.А. Молдовян, Н.А. Молдовян, Введение в криптосистемы с открытым ключом, БХВ-Петербург, 2005, с. 286 http://cyberdoc.nnm.ru/vvedenie\_v\_kriptosistemy\_s\_otkrytym\_klyuchom
12. А.Г.Ростовцев. Алгебраические основы криптографии, СПб, Мир и Семья, 2000.
13. А.Г.Ростовцев, Е.Б.Маховенко. Теоретическая криптография . – СПб.: АНО, ПО “Профессионал”, 2005,http://bookpedia.ru/index.php?newsid=1265
14. Г.Семенов. «Цифровая подпись. Эллиптические кривые».

<http://www.morepc.ru/security/crypt/os200207010.html?print>

14. В.Столлингс. Основы защиты сетей. Приложения и стандарты, М.: Вильямс, 2002, 429 с.

1. Брюс Шнайер. Прикладная криптография, 2-е издание: протоколы, алгоритмы и исходные тексты на языке С, http://www.ssl.stu.neva.ru/psw/crypto/appl\_rus/appl\_cryp.htm
2. Dr. Michael Ganley, Thales eSecurity Ltd. Метод эллиптических кривых, <http://www.racal.ru/rsp/eliptic_curve_cryptography.htm>
3. В.М.Фомичев. Дискретная математика и криптология, Диалог-МИФИ, 2003, 399 с.
4. Сайт Криптографический ликбез - <http://www.ssl.stu.neva.ru/psw/crypto.html>
5. Jovan Dj. Golic. Cryptanalysis of Alleged A5 Stream Cipher, Beograd, Yugoslavia, <http://jya.com/a5-hack.htm>
6. Ю. Спектор. Использование инструментов криптографии в Delphi-приложениях, <http://www.delphikingdom.com/asp/viewitem.asp?catalogID=1271>