Учреждение образования
«Минский государственный колледж электроники»

УТВЕРЖДЕНО
Заместитель директора
по учебной работе
_____Е.В.Филипцова
«____»_____20__ г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Защита компьютерной информации

(название учебной дисциплины)

ПРАКТИЧЕСКИЙ РАЗДЕЛ

для специальности (направления специальности) 2-40 01 01 «Программное обеспечение информационных технологий»

(код и наименование специальности (направления специальности, специализации)

Составители: Артемьева Е.А.

Рассмотрено на заседании цикловой комиссии по специальности 2-40 01 01 «Программное обеспечение информационных технологий»

Протокол № _____
«___»_____20__г.

Председатель цикловой комиссии Шавейко А.А./_____

Содержание

Лабораторная работа №1. Реализация генератора паролей с заданным параметром

Цель работы: реализовать простейший генератор паролей, обладающий требуемой стойкостью ко взлому.

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР1_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Подсистемы идентификации и аутентификации пользователя играют очень важную роль в системах защиты информации.

Стойкость подсистемы идентификации и аутентификации пользователя в системе защиты информации (СЗИ) во многом определяет устойчивость к взлому самой СЗИ. Данная стойкость определяется гарантией того, что злоумышленник не сможет пройти аутентификацию, присвоив чужой идентификатор или украв его.

Парольные системы идентификации/аутентификации является одними из основных и наиболее распространенных в СЗИ методами пользовательской аутентификации. В данном случае, информацией, аутентифицирующей пользователя, является некоторый секретный пароль, известный только легальному пользователю.

Парольная аутентификация пользователя является, как правило, передним краем обороны СЗИ. В связи с этим, модуль аутентификации по паролю наиболее часто подвергается атакам со стороны злоумышленника. Цель злоумышленника в данном случае – подобрать аутентифицирующую информацию (пароль) легального пользователя.

Методы парольной аутентификации пользователя являются наиболее простыми методами аутентификации и при несоблюдении определенных требований к выбору пароля являются достаточно уязвимыми.

Основными минимальными требованиями к выбору пароля и к подсистеме парольной аутентификации пользователя являются следующие.

<u>К паролю:</u>

1. Минимальная длина пароля должна быть не менее 6 символов.
2. Пароль должен состоять из различных групп символов (малые и большие латинские буквы, цифры, специальные символы ‘(’, ‘)’, ‘#’ и т.д.).
3. В качестве пароля не должны использоваться реальные слова, имена, фамилии и т.д.

<u>К подсистеме парольной аутентификации:</u>

1. Администратор СЗИ должен устанавливать максимальный срок действия пароля, после чего, он должен быть сменен.
2. В подсистеме парольной аутентификации должно быть установлено ограничение числа попыток ввода пароля (как правило, не более 3).
3. В подсистеме парольной аутентификации должна быть установлена временная задержка при вводе неправильного пароля.

Как правило, для генерирования паролей в СЗИ, удовлетворяющих перечисленным требованиям к паролям, используются программы - автоматические генераторы паролей пользователей.

При выполнении перечисленных требований к паролям и к подсистеме парольной аутентификации, единственно возможным методом взлома данной подсистемы злоумышленником является прямой перебор паролей (brute forcing). В данном случае, оценка стойкости парольной защиты осуществляется следующим образом.

Количественная оценка стойкости парольной защиты

Пусть A – мощность алфавита паролей (количество символов, которые могут быть использованы при составлении пароля. Например, если пароль состоит только из малых английских букв, то A=26).

L – длина пароля.

$S = A^L$ - число всевозможных паролей длины L, которые можно составить из символов алфавита A.

V – скорость перебора паролей злоумышленником.

T – максимальный срок действия пароля.

Тогда, вероятность P подбора пароля злоумышленником в течении срока его действия V определяется по следующей формуле.

$$P = \frac{V*T}{S} = \frac{V*T}{A^L}$$

Эту формулу можно использовать в обратную сторону для решения следующей задачи:

ЗАДАЧА. Определить минимальные мощность алфавита паролей A и длину паролей L, обеспечивающих вероятность подбора пароля злоумышленником не более заданной P, при скорости подбора паролей V, максимальном сроке действия пароля T.

Данная задача имеет неоднозначное решение. При исходных данных V,T,P однозначно можно определить лишь нижнюю границу S^* числа всевозможных паролей. Целочисленное значение нижней границы вычисляется по формуле

$$S^* = \left[\frac{V*T}{P} \right] \quad (1)$$

где $[\ ]$ - целая часть числа, взятая с округлением вверх.

После нахождения нижней границы S^* необходимо выбрать такие A и L для формирования $S=A^L$, чтобы выполнялось неравенство (2).

$$S^* \le S = A^L \quad (2)$$

При выборе S, удовлетворяющего неравенству (2), вероятность подбора пароля злоумышленника (при заданных V и T) будет меньше, чем заданная P.

Необходимо отметить, что при осуществлении вычислений по формулам (1) и (2), величины должны быть приведены к одним размерностям.

Пример

Исходные данные – $P=10^{-6}$, T=7 дней = 1 неделя, V=10 паролей / минуту = 10*60*24*7=100800 паролей в неделю.

Тогда, $S^* = \left[\frac{10800*1}{10^{-6}} \right] = 108*10^8$.

Условию $S^* \le A^L$ удовлетворяют, например, такие комбинации A и L, как A=26, L=8 (пароль состоит из 8 малых символов английского алфавита), A=36, L=6 (пароль состоит из 6 символов, среди которых могут быть малые латинские буквы и произвольные цифры).

Выбор пароля пользователя

Большое значение при реализации СЗИ имеет реализация подсистемы идентификации и аутентификации пользователей. Как правило, на переднем крае обороны используются парольные подсистемы аутентификации пользователей. В данных подсистемах пользователь аутентифицируется по паролю, известному только ему и ни кому более.

Стойкость к взлому подсистемы парольной идентифика-ции/аутентификации во многом определяется тем, насколько правильно были сформированы пароли пользователей. При несоблюдении ряда требований к выбору паролей, данная стойкость в значительной степени уменьшается, и подсистема идентификации/аутентификации становится достаточно уязвима при правильно построенной атаке.

Ниже перечислены основные требования, которые должны быть учтены при выборе пароля пользователя.

Минимальная длина пароля должна быть не менее 6 символов. Сокращение длины пароля во многом повышает вероятность успешной атаки полным их перебором.

Пароль должен состоять из различных групп символов (малые и большие латинские буквы, цифры, специальные символы ‘(’, ‘)’, ‘#’ и т.д.). Использование одной конкретной группы символов при формировании пароля в значительной степени повышает вероятность успешной атаки по маске.

В качестве пароля не должны использоваться реальные слова, имена, фамилии и т.д. Использование в качестве паролей конкретных слов, имен в значительной степени повышает вероятность успешной атаки по словарю.

Иногда, генераторы паролей могут использовать при данном генерировании элементы, входящие в идентификатор пользователя (отдельные его символы, количество символов и т.д.). В отдельных вариантах, пароль может формироваться даже целиком из идентификатора на основе некоторого алгоритма. В последнем случае, заданному идентификатору пользователя ставится в соответствие единственный пароль, который формируется на основе идентификатора. Данный вариант формирования пароля используется во многих коммерческих программах, требующих регистрации пользователя (например, WinZip).

Например,

Идентификатор пользователя Vasilyev

Пароль 1Op(0Qp+

При этом, при формировании пароля 1Op(0Qp+ могут использоваться отдельные символы, входящие в идентификатор Vasilyev.

Практическая часть

Задание 1.

В таблице 1 найти для вашего варианта значения характеристик P,V,T.

Таблица 1

Вариант	*P*	*V*	*T*
1	10^{-4}	15 паролей/мин	2 недели
2	10^{-5}	3 паролей/мин	10 дней
3	10^{-6}	10 паролей/мин	5 дней
4	10^{-7}	11 паролей/мин	6 дней
5	10^{-4}	100 паролей/день	12 дней
6	10^{-5}	10 паролей/день	1 месяц
7	10^{-6}	20 паролей/мин	3 недели
8	10^{-7}	15 паролей/мин	20 дней
9	10^{-4}	3 паролей/мин	15 дней
10	10^{-5}	10 паролей/мин	1 неделя
11	10^{-6}	11 паролей/мин	2 недели
12	10^{-7}	100 паролей/день	10 дней
13	10^{-4}	10 паролей/день	5 дней
14	10^{-5}	20 паролей/мин	6 дней
15	10^{-6}	15 паролей/мин	12 дней
16	10^{-7}	3 паролей/мин	1 месяц

Вычислить по формуле (1) нижнюю границу S* для заданных P,V,T.

Выбрать некоторый алфавит с мощностью A и получить минимальную длину пароля L, при котором выполняется условие (2).

Реализовать программу – генератор паролей пользователей. Программа должна формировать случайную последовательность символов длины L, при этом должен использоваться алфавит из A символов.

Замечания:

При реализации программы могут быть полезны следующие функции

RANDOM(N) – возвращает случайное число $0 \le r < N$.

RANDOMIZE – сбрасывает начальное состояние датчика случайных чисел случайным образом.

CHR(X) – возвращает символ с ASCII кодом X. Коды различных групп символов приведены ниже.

Коды символов:

Коды английских символов : «A»=65,…,«Z»=90, «a»=97,…, «z» =122.

Коды цифр : «0» = 48, «9» = 57.

! - 33, “ – 34, # - 35, $ - 36, % - 37, & - 38, ‘ – 39, (- 40,) – 41, * - 42.

Коды русских символов : «А» - 128, … «Я» - 159, «а» - 160,…, «п» - 175, «р» - 224,…, «я» - 239.

Задание 2

В таблице 2 найти требования, которым должен удовлетворять генератор паролей, соответствующий Вашему варианту.

Написать программу-генератор паролей, в соответствие с требованиями Вашего варианта. Программа должна выполнять следующие действия:

Ввод идентификатора пользователя с клавиатуры. Данный идентификатор представляет собой последовательность символов $a_1 a_2 \ldots a_N$, где N – количество символов идентификатора (может быть любым), a_i - i – ый символ идентификатора пользователя.

1.4.2 Формирование пароля пользователя $b_1 b_2 \ldots b_M$ для данного идентификатора, где M – количество символов пароля, соответствующее Вашему варианту, и вывод его на экран. Алгоритм получения символов пароля b_i указан в перечне требований Таблицы 2 для Вашего варианта.

Таблица 2

Вариант	Количество символов пароля	Перечень требований
1	6	1. b_1, b_2 - случайные заглавные буквы английского алфавита. 2. $b_3 = N^2 \bmod 10$ (где mod 10 – остаток от деления числа на 10). 3. b_4 - случайная цифра. 4. b_5 - случайный символ из множества {!,",#,$,%,&,',(,),*}. 5. b_6 - случайная малая буква английского алфавита.
2	7	1. b_1, b_2, b_3 - случайные малые буквы английского алфавита. 2. b_4, b_5 - случайные заглавные буквы английского алфавита. 3. $b_6 b_7$ - двузначное число, равное $N^4 \bmod 100$. (Если остаток – однозначное число, то $b_6 = 0$).
3	8	1. b_1, b_2, b_3 - случайные цифры. 2. b_4, b_5 - случайные символы из множества {!,",#,$,%,&,',(,),*}. 3. b_7 - случайная заглавная буква английского алфавита. 4. b_8 - P –ая по счету малая буква английского алфавита, где $P = N^2 \bmod 10 + N^3 \bmod 10 + 1$.
4	9	1. $b_1, ..., b_{1+Q}$ - случайные символы из множества {!,",#,$,%,&,',(,),*}, где $Q = N \bmod 5$. 2. Оставшиеся символы пароля, кроме b_9, - случайные малые буквы английского алфавита. 3. b_9 - случайная цифра.
5	10	1. $b_{10-Q}, ..., b_{10}$ - случайные цифры, где $Q = N \bmod 6$. 2. b_1, b_2 - случайные большие буквы английского алфавита. 3. $b_3, ..., b_{10-Q-1}$ - случайные малые буквы английского алфавита.
6	11	1. b_1, b_2 - случайные цифры. 2. $b_3, ..., b_{3+Q}$ - случайные большие буквы английского алфавита, где $Q = N \bmod 8$. 3. $b_{4+Q}, ..., b_{11}$ - случайные символы из множества {!,",#,$,%,&,',(,),*}.
7	11	1. b_1, b_2 - случайные цифры. 2. $b_3, ..., b_{3+Q}$ - случайные малые буквы русского алфавита, где $Q = N \bmod 8$. 3. $b_{4+Q}, ..., b_{11}$ - случайные символы из множества {!,",#,$,%,&,',(,),*}.
8	12	1. $b_1, ..., b_{1+Q}$ - случайные малые буквы английского алфавита, где $Q = N^3 \bmod 5$. 2. $b_{1+Q+1}, ..., b_{1+Q+1+P}$ - случайные заглавные буквы английского алфавита, где $P = N^2 \bmod 6$. 3. Оставшиеся символы пароля – случайные цифры.

Продолжение таблицы 2

9	12	1. $b_1,...,b_{1+Q}$ - случайные малые буквы русского алфавита, где $Q = N^3 \bmod 5$. 2. $b_{1+Q+1},...,b_{1+Q+1+P}$ - случайные заглавные буквы русского алфавита, где $P = N^2 \bmod 6$. 3. Оставшиеся символы пароля – случайные цифры.
10	10	1. $b_{10-Q},...,b_{10}$ - случайные цифры, где $Q = N \bmod 6$. 2. b_1, b_2 - случайные большие буквы русского алфавита. 3. $b_3,...,b_{10-Q-1}$ - случайные малые буквы русского алфавита.
11	9	1. $b_1, b_2 ... b_{1+Q}$ - случайные символы из множества {!,",#,$,%,&,',(,),*}, где $Q = N \bmod 5$. 2. Оставшиеся символы пароля, кроме b_9, - случайные малые буквы русские алфавита. 3. b_9 - случайная цифра.
12	8	1. b_1, b_2, b_3 - случайные цифры. 2. b_4, b_5 - случайные символы из множества {!,",#,$,%,&,',(,),*}. 3. b_7 - случайная заглавная буква русского алфавита. 4. b_8 - P –ая по счету малая буква русского алфавита, где $P = N^2 \bmod 15 + N^3 \bmod 15 + 1$.
13	7	1. b_1, b_2, b_3 - случайные малые буквы русского алфавита. 2. b_4, b_5 - случайные заглавные буквы русского алфавита. 3. $b_6 b_7$ - двузначное число, равное $N^4 \bmod 100$. (Если остаток – однозначное число, то $b_6 = 0$).
14	6	1. b_1, b_2 - случайные заглавные буквы русского алфавита. 2. $b_3 = N^2 \bmod 10$ (где mod 10 – остаток от деления числа на 10). 3. b_4 - случайная цифра. 4. b_5 - случайный символ из множества {!,",#,$,%,&,',(,),*}. 5. b_6 - случайная малая буква русского алфавита.
15	6	1. b_1, b_2 - случайные заглавные буквы английского алфавита. 2. $b_3 = N^2 \bmod 10$ (где mod 10 – остаток от деления числа на 10). 3. b_4 - случайная цифра. 4. b_5 - случайный символ из множества {!,",#,$,%,&,',(,),*}. 5. b_6 - случайная малая буква русского алфавита.
16	7	1. b_1, b_2, b_3 - случайные малые буквы русского алфавита. 2. b_4, b_5 - случайные заглавные буквы английского алфавита. 3. $b_6 b_7$ - двузначное число, равное $N^4 \bmod 100$. (Если остаток – однозначное число, то $b_6 = 0$).

ЗАМЕЧАНИЯ

1.Коды английских символов - «A»=65,…,«Z»=90, «a»=97,…, «z» =122.

2.Коды цифр – «0» = 48, «9» = 57.

3.Коды спец. символов ! – 33, “ – 34, # - 35, $ - 36, % - 37, & - 38, ‘ – 39, (- 40,) – 41, * - 42.

4.Коды русских символов – «А» - 128, … «Я» - 159, «а» - 160,…, «п» - 175, «р» - 224,…, «я» - 239.

Контрольные вопросы:

1. В чем преимущество программных генераторов паролей по сравнению с выбором паролей человеком (пользователем либо администратором)?
2. Желательно либо нежелательно, по Вашему мнению, генерирование пароля пользователя на основании некоторого алгоритма из его идентификатора? Повысится либо понизится стойкость защиты при использовании такого алгоритма?
3. Чем определяется стойкость подсистемы идентификации и аутентификации?
4. Перечислить минимальные требования к выбору пароля.
5. Перечислить минимальные требования к подсистеме парольной аутентификации.
6. Как определить вероятность подбора пароля злоумышленником в течении срока его действия?
7. Выбором каким параметров можно повлиять на уменьшение вероятности подбора пароля злоумышленником при заданной скорости подбора пароля злоумышленником и заданном сроке действия пароля?

Лабораторная работа № 2. Шифрование методами перестановки

Цель работы: закрепить умение шифровать информацию методом шифрующих таблиц и методом магического квадрата.

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР2_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Шифрующие таблицы основаны на заполнении их ячеек буквами текста. Ключом шифрующих таблиц может быть:

- размер таблицы (число строк и столбцов);
- слово или фраза, определяющая перестановку строк и (или) столбцов.

Шифрующие таблицы

Идея метода простых шифрующих таблиц придумана задолго до появления современной цивилизации и предельно проста. Пусть известны величины m - число строк, эта цифра является ключом. Если длина открытого текста не кратна m, то придётся дополнить текст чем-нибудь, чтобы избавиться от этого свойства. Например, пробелами или не значащими буквами. Так, если входные данные состоят из 26 символов, m = 3, то добавляем 1 пробел или букву.

Результат записываем по столбцам слева направо, для понятности пример: 3 строки текст "демократы"

Д	О	А
Е	К	Т
М	Р	Ы

А теперь считываем по строкам сверху вниз. В нашем примере получим недемократичное слово "ДОАЕКТМРЫ". Таким способом шифруется весь текст, при этом пробелы не имеют никаких привилегий и считаются обычными символами.

Результат объединяем в один текст - шифрованный.

Есть смысл стараться брать длину блока m такой, чтобы число m*n (n- количество столбцов) имело как можно больше делителей. Это увеличивает количество вариантов шифрующих таблиц при известной длине блока. Например, 25 - плохая длина блока: есть только вариант 5*5, 24 - получше - есть варианты 2*12, 3*8, 4*6, 6*4, 8*3, 12*2. Как видим, для небольших чисел m, n метод простых шифрующих таблиц легко вскрывается даже вручную. Хоть сколько-нибудь солидное количество вариантов ключа в методе простых шифрующих таблиц возможно только для очень длинных блоков.

Магический квадрат

Магическими квадратами называются квадратные таблицы со вписанными в их клетки последовательными натуральными числами от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число. Подобные квадраты широко применялись для вписывания шифруемого текста по приведенной в них нумерации. Если потом выписать содержимое таблицы по строкам, то получалась шифровка перестановкой букв. На первый взгляд кажется, будто магических квадратов очень мало. Тем не менее, их число очень быстро возрастает с увеличением размера квадрата. Так, существует лишь один магический квадрат размером 3 x 3, если не принимать во внимание его повороты. Магических квадратов 4 x 4 насчитывается уже 880, а число магических квадратов размером 5 x 5 около 250000. Поэтому магические квадраты больших размеров могли быть хорошей основой для надежной системы шифрования того времени, потому что ручной перебор всех вариантов ключа для этого шифра был немыслим.

В квадрат размером 4 на 4 вписывались числа от 1 до 16. Его магия состояла в том, что сумма чисел по строкам, столбцам и полным диагоналям равнялась одному и тому же числу — 34. Впервые эти квадраты появились в Китае, где им и была приписана некоторая «магическая сила».

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Шифрование по магическому квадрату производилось следующим образом. Например, требуется зашифровать фразу: «ПриезжаюСегодня.». Буквы этой фразы вписываются последовательно в квадрат согласно записанным в них числам: позиция буквы в предложении соответствует порядковому числу. В пустые клетки ставится точка.

16.	3 и	2 р	13 д
5 з	10 е	11 г	8 ю
9 С	6 ж	7 а	12 о
4 е	15 я	14 н	1 П

После этого шифрованный текст записывается в строку (считывание производится слева направо, построчно):

.ирдзегюСжаоеянП

При расшифровывании текст вписывается в квадрат, и открытый текст читается в последовательности чисел «магического квадрата». Программа должна генерировать «магические квадраты» и по ключу выбирать необходимый. Размер квадрата больше чем 3x3.

Практическая часть

Задание №1

Зашифровать методом шифрующих таблиц индивидуальную фразу. Попытаться выбрать максимально криптостойкую таблицу.

В качестве индивидуальной фразы придумать «сообщение себе в прошлое».

Задание №2

Зашифровать методом магического квадрата индивидуальную фразу. Попытаться выбрать максимально криптостойкую таблицу.

В качестве индивидуальной фразы придумать «сообщение себе же в будущее».

Выполнить шифрование 2-х фраз для 2-х видов квадратов (5x5, 6x6)

21	24	2	3	15
1	6	16	22	20
14	12	19	7	13
25	5	17	10	8
4	18	11	23	9

2	18	1	23	21
12	25	5	4	19
16	9	15	14	11
13	3	24	17	8
22	10	20	7	6

4	24	10	15	12
25	13	14	6	7
3	18	22	20	2
17	9	11	5	23
16	1	8	19	21

22	36	7	2	9	35
26	18	31	10	5	21
13	23	15	24	28	8
12	4	14	34	30	17
6	1	33	25	19	27
32	29	11	16	20	3

18	28	3	12	15	35
32	11	14	17	4	33
20	9	24	13	16	29
21	27	10	25	23	5
1	30	34	8	31	7
19	6	26	36	22	2

8	10	24	4	32	33
29	20	28	21	1	12
36	5	22	14	3	31
2	27	18	30	25	9
17	26	6	35	16	11
19	23	13	7	34	15

Контрольные вопросы:

1. Что называется ключом в шифрующих таблицах?
2. Опишите метод шифрующих таблиц.
3. Опишите метод магического квадрата.

Лабораторная работа №3 Шифрование методами замены

Цель работы: закрепить умение шифровать информацию системой Цезаря, системой Трисемуса, алгоритмом Плейфера и реализовать алгоритмы в программном коде

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР3_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Система 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.

Математическая модель

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

$$y = x + k \ (mod \ n)$$

$$x = y - k \ (mod \ n),$$

где x — символ открытого текста

y — символ шифрованного текста

n — мощность алфавита (кол-во символов)

k — ключ.

Алфавит:

Буква	**А**	**Б**	**В**	**Г**	**Д**	**Е**	**Ё**	**Ж**	**З**	**И**	**Й**
Номер	1	2	3	4	5	6	7	8	9	10	11
Буква	**К**	**Л**	**М**	**Н**	**О**	**П**	**Р**	**С**	**Т**	**У**	**Ф**
Номер	12	13	14	15	16	17	18	19	20	21	22
Буква	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ь**	**Ы**	**Ъ**	**Э**	**Ю**	**Я**
Номер	23	24	25	26	27	28	29	30	31	32	33

Пример:

Сообщение	К	Р	И	П	Т	О	Г	Р	А	Ф	И	Я
Номер 1	12	18	10	17	20	16	4	18	1	22	10	33
Номер 1 +5	17	23	15	22	25	21	9	23	6	27	15	5
Шифр	П	Х	Н	Ф	Ч	У	З	Х	Е	Щ	Н	Д

Ответ: «Пхнфчузхещнд» Ключ: 5

Система Трисемуса

Шифрующая система Трисемуса (Тритемия). В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием «Полиграфия». В этой книге он впервые систематически описал применение модифицированного шифра Цезаря.

Здесь шаг смещения делается переменным, то есть зависящим от каких-либо дополнительных факторов. Например, можно задать закон смещения в виде линейной функции (уравнения зашифрования) позиции шифруемой буквы.

Сама функция должна гарантировать целочисленное значение. Прямая функция шифрования должна иметь обратную функцию шифрования, тоже целочисленную.

Уравнение зашифрования для шифра Тритемиуса имеет следующий вид:

$$L=(m+k)\bmod N$$

Некоторые варианты вычисления шага смещения k:

k=A*p+B,

k=A*p^2+B*p+C,

где p — позиция буквы в сообщении; A, B, C — ключи.

Алгоритм шифрования с использованием системы Трисемуса:

1. Определяем порядковый номер шифруемой буквы в тексте.
2. Определяем код буквы в алфавите.
3. Вычисляем смещение k.
4. Находим код зашифрованной буквы, пользуясь нашим уравнением зашифрования.L=(m+k)mod N Расшифрование m=(L-k)mod N
5. По коду L восстанавливаем очередную букву криптограммы.
6. Повторяем пункты 1..5 до окончания текста шифрограммы.

Для k=2p^2+5p+3 и алфавита:

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я		,	.
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Оригинальный текст:

Съешь же ещё этих мягких французских булок, да выпей чаю.

Шифрованный текст:

ФБЩШЛГД Ч.ЪСЧДП ЕО,ЧЁЬЙЙЛЮЩЛ РЬА РЙХАКЕЛ,РЮШЮЭ,НТЩВ,ПЁФЦВ

Реализация алгоритма шифрования Плейфера

Шифр Плейфера использует матрицу 5x5 (для латинского алфавита, для кириллического алфавита необходимо увеличить размер матрицы до 4x8), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки матрицы буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо, либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом, составляет матрицу 5x5 и является ключом шифра.

Для того чтобы зашифровать сообщение, необходимо разбить его на биграммы (группы из двух символов), например «Hello World» становится «HE LL OW OR LD», и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга. Затем, руководствуясь следующими 4 правилами, зашифровываем пары символов исходного текста:

1. Если два символа биграммы совпадают (или если остался один символ), добавляем после первого символа «X», зашифровываем новую пару символов и продолжаем. В некоторых вариантах шифра Плейфера вместо «X» используется «Q».

2. Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

3. Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящиеся непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

4. Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

Для расшифровки необходимо использовать инверсию этих четырёх правил, откидывая символы «X» (или «Q»), если они не несут смысла в исходном сообщении.

Пример. Используем ключ «playfair example», тогда матрица примет вид:

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
J	K	N	O	S
T	U	V	W	Z

Зашифруем сообщение «Hide the gold in the tree stump»

HI DE TH EG OL DI NT HE TR EX ES TU MP

1. Биграмма HI формирует прямоугольник, заменяем её на BM.
2. Биграмма DE расположена в одном столбце, заменяем её на ND.
3. Биграмма TH формирует прямоугольник, заменяем её на ZB.
4. Биграмма EG формирует прямоугольник, заменяем её на XD.
5. Биграмма OL формирует прямоугольник, заменяем её на KY.
6. Биграмма DI формирует прямоугольник, заменяем её на BE.
7. Биграмма NT формирует прямоугольник, заменяем её на JV.
8. Биграмма HE формирует прямоугольник, заменяем её на DM.
9. Биграмма TR формирует прямоугольник, заменяем её на UI.
10. Биграмма EX находится в одной строке, заменяем её на XM.
11. Биграмма ES формирует прямоугольник, заменяем её на MN.
12. Биграмма TU находится в одной строке, заменяем её на UV.
13. Биграмма MP формирует прямоугольник, заменяем её на IF.

Получаем зашифрованный текст «BM ND ZB XD KY BE JV DM UI XM MN UV IF»

Таким образом сообщение «Hide the gold in the tree stump» преобразуется в «BMNDZBXDKYBEJVDMUIXMMNUVIF»

Практическая часть

Задание №1

Зашифровать системой Цезаря индивидуальную фразу.

В качестве индивидуальной фразы зашифровать *«Свое Имя Отчество Фамилию».*

В качестве ключа использовать свой номер в списке.

Задание №2

Зашифровать методом Трисемуса индивидуальную фразу.

В качестве индивидуальной фразы использовать «Название предмета который мы проходим».

В качестве коэффициентов использовать цифры из таблицы

#	A	B	C
1	1	4	-3
2	2	-1	2
3	3	2	1
4	2	5	2
5	4	1	4
6	2	-1	7
7	3	3	-2
8	1	4	2
9	2	2	3
10	1	5	1
11	2	5	2
12	6	1	4
13	2	-1	7
14	3	4	-5
15	1	4	2
16	2	1	3

Задание №3

Зашифровать системой Плейфера индивидуальную фразу.

В качестве индивидуальной фразы зашифровать «Название любимой книги и ее автора».

В качестве ключа использовать слово из таблицы.

1 вариант: буйвол	9 вариант: мгновенье
2 вариант: мальта	10 вариант: гвардия
3 вариант: портрет	11 вариант: колонна
4 вариант: архив	12 вариант: сплав
5 вариант: совершенство	13 вариант: призыв
6 вариант: удар	14 вариант: шерсть
7 вариант: алгоритм	15 вариант: фамилия
8 вариант: советник	16 вариант: парус

Контрольные вопросы:

1. Опишите шифр Цезаря.
2. Опишите алгоритм системы Трисемуса.
3. Как зашифровывается сообщение в шифре Плейфера?

Лабораторная работа № 4. Реализация генератора псевдослучайной последовательности

Цель работы: изучить и закрепить умение генерирования алгоритмов работы генераторов псевдослучайных чисел.

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР4_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Способы получения случайных чисел

В программировании достаточно часто находят применение последовательности чисел, выбранных случайным образом из некоторого множества. В качестве примеров задач, в которых используются случайные числа, можно привести следующие:

- тестирование алгоритмов;
- имитационное моделирование;
- некоторые задачи численного анализа;
- имитация пользовательского ввода.

Для получения случайных чисел можно использовать различные способы. В общем случае все методы генерирования случайных чисел можно разделить на аппаратные и программные. Устройства или алгоритмы получения случайных чисел называют генераторами случайных чисел (ГСЧ) или датчиками случайных чисел.

Аппаратные ГСЧ представляют собой устройства, преобразующие в цифровую форму какой-либо параметр окружающей среды или физического процесса. Параметр и процесс выбираются таким образом, чтобы обеспечить хорошую «случайность» значений при считывании. Очень часто используются паразитные процессы в электронике (токи утечки, туннельный пробой диодов, цифровой шум видеокамеры, шумы на микрофонном входе звуковой карты и т.п.). Формируемая таким образом последовательность чисел, как правило, носит абсолютно случайный характер и не может быть воспроизведена заново по желанию пользователя.

К программным ГСЧ относятся различные алгоритмы генерирования последовательности чисел, которая по своим характеристикам напоминает случайную. Для формирования очередного числа последовательности используются различные алгебраические преобразования. Одним из первых программных ГСЧ является метод средин квадратов, предложенный в 1946 г. Дж. фон Нейманом.

Любые программные ГСЧ, не использующие внешних «источников энтропии» и формирующие очередное число только алгебраическими преобразованиями, не дают чисто случайных чисел. Последовательность на выходе такого ГСЧ выглядит как случайная, но на самом деле подчиняется некоторому закону и, как правило, рано или поздно зацикливается. Такие числа называются псевдослучайными.

Случайное число – число, представляющее собой реализацию случайной величины.

Детерминированный алгоритм – алгоритм, который возвращает те же выходные значения при тех же входных значениях.

Псевдослучайное число – число, полученное детерминированным алгоритмом, используемое в качестве случайного числа.

Физическое случайное число (истинно случайное) – случайное число, полученное на основе некоторого физического явления.

Генератор псевдослучайных чисел — алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

Линейный конгруэнтный генератор псевдослучайных чисел

Генераторы псевдослучайных чисел могут работать по разным алгоритмам. Одним из простейших генераторов является так называемый линейный конгруэнтный генератор, который для вычисления очередного числа ki использует формулу

ki=(a*ki-1+b)mod c,

где a, b, c — некоторые константы, а ki-1 — предыдущее псевдослучайное число. Для получения k1 задается начальное значение k0. Возьмем в качестве примера a=5,b=3,c=11 и пусть k0= 1. В этом случае мы сможем по приведенной выше формуле получать значения от 0 до 10 (так как c = 11). Вычислим несколько элементов последовательности:

k1 = (5 * 1 + 3) mod 11 = 8;

k2 = (5 * 8 + 3) mod 11 = 10;

k3 = (5 * 10 + 3) mod 11 = 9;

k4 = (5 * 9 + 3) mod 11 = 4;

k5 = (5 * 4 + 3) mod 11 = 1.

Полученные значения (8, 10, 9, 4, 1) выглядят похожими на случайные числа. Однако следующее значение k6 будет снова равно 8:

k6 = (5 * 1 + 3) mod 11 = 8,

а значения k7 и k8 будут равны 10 и 9 соответственно:

k7 = (5 * 8 + 3) mod 11 = 10;

k8= (5 * 10 + 3) mod 11 = 9.

Выходит, наш генератор псевдослучайных чисел повторяется, порождая периодически числа 8, 10, 9, 4, 1. К сожалению, это свойство характерно для всех линейных конгруэнтных генераторов. Изменяя значения основных параметров a, b и c, можно влиять на длину периода и на сами порождаемые значения ki. Так, например, увеличение числа c в общем случае ведет к увеличению периода. Если параметры a, b и c выбраны правильно, то генератор будет порождать случайные числа с максимальным периодом, равным c. При программной реализации значение c обычно устанавливается равным 2b-1 или 2b, где b — длина слова ЭВМ в битах.

Достоинством линейных конгруэнтных генераторов псевдослучайных чисел является их простота и высокая скорость получения псевдослучайных значений. Линейные конгруэнтные генераторы находят применение при решении задач моделирования и математической статистики, однако в криптографических целях их нельзя рекомендовать к использованию, так как специалисты по криптоанализу научились восстанавливать всю последовательность ПСЧ по нескольким значениям. Например, предположим, что противник может определить значения k0, k1, k2, k3. Тогда:

k1=(a*k0+b) mod c

k2=(a*k1+b) mod c

k3=(a*k2+b) mod c

Решив систему из этих трех уравнений, можно найти a, b и c.

Для получения псевдослучайных чисел предлагалось использовать также квадратичные и кубические генераторы:

ki=(a12*ki-1+a2*ki-1+b) mod c

ki=(a13*ki-1+a22*ki-1+a3*ki-1+b) mod c

Однако такие генераторы тоже оказались непригодными для целей криптографии по той же самой причине "предсказуемости".

Метод Фибоначчи с запаздыванием

Метод Фибоначчи с запаздываниями (Lagged Fibonacci Generator) — один из методов генерации псевдослучайных чисел. Он позволяет получить более высокое "качество" псевдослучайных чисел.

Наибольшую популярность фибоначчиевы датчики получили в связи с тем, что скорость выполнения арифметических операций с вещественными числами сравнялась со скоростью целочисленной арифметики, а фибоначчиевы датчики естественно реализуются в вещественной арифметике.

Известны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространённых фибоначчиевых датчиков основан на следующей рекуррентной формуле:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & если\ k_{i-a} \geq k_{i-b} \\ k_{i-a} - k_{i-b} + 1, & если\ k_{i-a} < k_{i-b} \end{cases}$$

где ki — вещественные числа из диапазона [0,1], a, b — целые положительные числа, параметры генератора. Для работы фибоначчиеву датчику требуется знать max{a,b} предыдущих сгенерированных случайных чисел. При программной реализации для хранения сгенерированных случайных чисел необходим некоторый объем памяти, зависящих от параметров a и b.

Пример. Вычислим последовательность из первых десяти чисел, генерируемую методом Фибоначчи с запаздыванием начиная с k5 при следующих исходных данных: a = 4, b = 1, k0=0.1; k1=0.7; k2=0.3; k3=0.9; k4=0.5:

k5 = k1 - k4 = 0.7 - 0.5 = 0.2;
k6 = k2 - k5= 0.3 - 0.2 = 0.1;
k7 = k3 - k6 = 0.9 - 0.1 = 0.8;
k8 = k4 - k7 + 1 =0.5 - 0.8 + 1 = 0.7;
k9 = k5- k8 + 1 =0.2 - 0.7 + 1 = 0.5;
k10 = k6 - k9 + 1 =0.1 - 0.5 + 1 = 0.6;
k11 = k7 - k10 = 0.8 - 0.6 = 0.2;
k12 = k8 - k11 = 0.7 - 0.2 = 0.5;
k13 = k9 - k12 + 1 =0.5 - 0.5 + 1 = 1;
k14 = k10 - k13 + 1 =0.6 - 1 + 1 = 0.6.

Видим, что генерируемая последовательность чисел внешне похожа на случайную. И действительно, исследования подтверждают, что получаемые случайные числа обладают хорошими статистическими свойствами.

Для генераторов, построенных по методу Фибоначчи с запаздыванием, существуют рекомендуемые параметры a и b, так сказать, протестированные на качество. Например, исследователи предлагают следующие значения: (a,b) = (55, 24), (17, 5) или (97,33). Качество получаемых случайных чисел зависит от значения константы a: чем оно больше, тем выше размерность пространства, в котором сохраняется равномерность случайных векторов, образованных из полученных случайных чисел. В то же время с увеличением величины константы a увеличивается объём используемой алгоритмом памяти.

В результате значения (a,b) = (17,5) рекомендуются для простых приложений. Значения (a,b) = (55,24) позволяют получать числа, удовлетворительные для большинства криптографических алгоритмов, требовательных к качеству случайных чисел. Значения (a,b) = (97,33) позволяют получать очень качественные случайные числа и используются в алгоритмах, работающих со случайными векторами высокой размерности.

Генераторы ПСЧ, основанные на методе Фибоначчи с запаздыванием, использовались для целей криптографии. Кроме того, они применяются в математических и статистических расчетах, а также при моделировании случайных процессов. Генератор ПСЧ, построенный на основе метода Фибоначчи с запаздыванием, использовался в широко известной системе Matlab.

Практическая часть

Задание№1

Выдайте на экран 10 случайных равномерно распределенных чисел в диапазоне:

- ✓ От 3 до 12, целые.
- ✓ Из множества {–3, 0, 6, 9, 12, 15}.
- ✓ От 3 до 12, вещественные.
- ✓ От –2,3 до 10,7 с шагом 0,1.
- ✓ Из множества {–30; 10; 63; 59; 120; 175}.
- ✓ Из множества {1; 0,1; 0,01; …; 10^{-15}}.

Задание№2

Напишите программу, моделирующую игру «Быки и коровы». Программа выбирает с помощью датчика случайных чисел четырехзначное число с разными цифрами. Цель игры – угадать это число. На каждом шаге играющий называет четырехзначное число, а программа сообщает, сколько цифр числа угадано (быки) и сколько угаданных цифр стоит на нужном месте (коровы).

Контрольные вопросы:

1. Случайное число представляет собой….
2. Детерминированный алгоритм – это…
3. Псевдослучайное число – это…
4. Физическое случайное число – это…
5. Генератор псевдослучайных чисел – это…
6. Опишите линейный конгруэнтный генератор псевдослучайных чисел.
7. Опишите метод Фибоначчи с запаздыванием.

Лабораторная работа № 5. Исследование симметричных криптоалгоритмов

Цель: Закрепить сведения о различных симметричных алгоритмах шифрования и реализовать их в программном коде

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР5_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретические сведенья

Шифрующие таблицы

С начала эпохи Возрождения (конец XIV столетия) начала возрождаться и криптография. Наряду с традиционными применениями криптографии в политике, дипломатии и военном деле появляются и другие задачи - защита интеллектуальной собственности от преследований инквизиции или заимствований злоумышленников. В разработанных шифрах перестановки того времени применяются шифрующие таблицы, которые в сущности задают правила перестановки букв в сообщении.

В качестве ключа в шифрующих таблицах используются:

- размер таблицы;
- слово или фраза, задающие перестановку;
- особенности структуры таблицы.

Одним из самых примитивных табличных шифров перестановки является простая перестановка, для которой ключом служит размер таблицы. Этот метод шифрования сходен с шифром *скитала*. Например, сообщение

ТЕРМИНАТОР ПРИБЫВАЕТ СЕДЬМОГО В ПОЛНОЧЬ

записывается в таблицу поочередно по столбцам. Результат заполнения таблицы из 5 строк и 7 столбцов показан на рисунке 2.

После заполнения таблицы текстом сообщения по столбцам для формирования шифртекста считывают содержимое таблицы по строкам.

Т	Н	П	В	Е	Г	Л
Е	А	Р	А	Д	О	Н
Р	Т	И	Е	Ь	В	О
М	О	Б	Т	М	П	Ч
И	Р	Ы	С	О	О	Ь

Рисунок 2 Заполнение таблицы из 5 строк и 7 столбцов

Если шифртекст записывать группами по пять букв, получается такое шифрованное сообщение:

ТНПВЕ ГЛЕАР АДОНР ТИЕЬВ ОМОБТ МПЧИР ЫСООЬ

Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы. Следует заметить, что объединение букв шифртекста в 5-буквенные группы не входит в ключ шифра и осуществляется для удобства записи несмыслового текста. При расшифровании действия выполняют в обратном порядке.

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый одиночной перестановкой по ключу. Этот метод отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы.

Применим в качестве ключа, например, слово

ПЕЛИКАН,

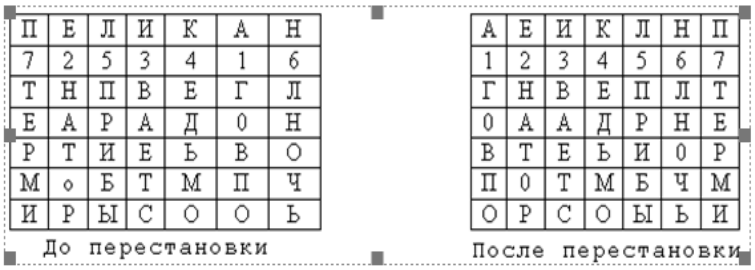


Рисунок 3 Таблицы, заполненные ключевым словом и текстом сообщения

а текст сообщения возьмем из предыдущего примера. На рисунке 3 показаны две таблицы, заполненные текстом сообщения и ключевым словом, при этом левая таблица соответствует заполнению до перестановки, а правая таблица - заполнению после перестановки.

В верхней строке левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с естественным порядком соответствующих букв ключа в алфавите. Если бы в ключе встретились одинаковые буквы, они бы были понумерованы слева направо. В правой таблице столбцы переставлены в соответствии с упорядоченными номерами букв ключа.

При считывании содержимого правой таблицы по строкам и записи шифртекста группами по пять букв получим шифрованное сообщение:

ГНВЕП ЛТООА ДРНЕВ ТЕЬИО РПОТМ БЧМОР СОЫЬИ

Для обеспечения дополнительной скрытности можно повторно зашифровать сообщение, которое уже прошло шифрование. Такой метод шифрования называется *двойной перестановкой*. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При расшифровании порядок перестановок должен быть обратным.

Пример выполнения шифрования методом двойной перестановки показан на рисунке 4. Если считывать шифртекст из правой таблицы построчно блоками по четыре буквы, то получится следующее:

ТЮАЕ ООГМ РЛИП ОЬСВ

Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы (в нашем примере последовательности 4132 и 3142 соответственно).

Рисунок 4 Пример выполнения шифрования методом двойной перестановки

Число вариантов двойной перестановки быстро возрастает при увеличении размера таблицы:

- для таблицы 3x3 36 вариантов;
- для таблицы 4x4 576 вариантов;
- для таблицы 5x5 14400 вариантов.

Однако двойная перестановка не отличается высокой стойкостью и сравнительно просто «взламывается» при любом размере таблицы шифрования.

Квадрат Полибия

Шаг 1: Формирование таблицы шифрования

К каждому языку отдельно составляется таблица шифрования с одинаковым (не обязательно) количеством пронумерованных строк и столбцов, параметры которой зависят от его мощности (количества букв в алфавите). Берутся два целых числа, произведение которых ближе всего к количеству букв в языке — получаем нужное число строк и столбцов. Затем вписываем в таблицу все буквы алфавита подряд — по одной на каждую клетку. При нехватке клеток можно вписать в одну две буквы (редко употребляющиеся или схожие по употреблению).

Латинский алфавит

В современном латинском алфавите 26 букв, следовательно таблица должна состоять из 5 строк и 5 столбцов, так как 25=5*5 наиболее близкое к 26 число. При этом буквы I, J не различаются (J отождествляется с буквой I), так как не хватает 1 ячейки:

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Русский алфавит

Идею формирования таблицы шифрования проиллюстрируем для русского языка. Число букв в русском алфавите отличается от числа букв в греческом алфавите, поэтому размер таблицы выбран другой (квадрат 6*6=36, поскольку 36 наиболее близкое число к 33):

	1	**2**	**3**	**4**	**5**	**6**
1	А	Б	В	Г	Д	Е
2	Ё	Ж	З	И	Й	К
3	Л	М	Н	О	П	Р
4	С	Т	У	Ф	Х	Ц
5	Ч	Ш	Щ	Ъ	Ы	Ь
6	Э	Ю	Я	—	—	—

Возможен также другой вариант составления, предусматривающий объединение букв Е и Ё, И и Й, Ъ и Ь. В данном случае получаем следующий результат:

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е/Ё
2	Ж	З	И/Й	К	Л	М
3	Н	О	П	Р	С	Т
4	У	Ф	Х	Ц	Ч	Ш
5	Щ	Ы	Ь/Ъ	Э	Ю	Я

Используя подобный алгоритм, таблицу шифрования можно задать для любого языка. Чтобы расшифровать закрытый текст необходимо знать, таблицей шифрования какого алфавита он зашифрован.

Или есть такой вариант: Шифр «Квадрат Полибия».

«Квадрат Полибия» представляет собой квадрат 5×5, столбцы и строки которого нумеруются цифрами от 1 до 5. В каждую клетку этого квадрата записывается одна буква (в нашем алфавите 31 буква, Ъ и Ё исключены, кроме того в одну клетку поместите буквы е-э, и-й, ж-з, р-с, ф-х, ш-щ). Буквы расположены в алфавитном порядке. В результате каждой букве соответствует пара чисел, и шифрованное сообщение превращается в последовательность пар чисел. Расшифровывается путем нахождения буквы, стоящей на пересечении строки и столбца.

	1	2	3	4	5
1	А	Б	В	Г	Д
2	Е/Э	Ж	З	И/Й	К
3	Л	М	Н	О	П
4	Р/С	Т	У	Ф/Х	Ц
5	Ч	Ш/Щ	Ы	Ю	Я

Шаг 2: Принцип шифрования

Существует несколько методов шифрования с помощью квадрата Полибия. Ниже приведены три из них.

Метод 1

Сообщение преобразуется в координаты по квадрату Полибия, координаты записываются вертикально:

Таблица координат								
Буква:	**S**	**O**	**M**	**E**	**T**	**E**	**X**	**T**
Координата горизонтальная:	**3**	**4**	**2**	**5**	**4**	**5**	**3**	**4**
Координата вертикальная:	**4**	**3**	**3**	**1**	**4**	**1**	**5**	**4**

Затем координаты считывают по строкам:

34 25 45 34 43 31 41 54 (*)

Далее координаты преобразуются в буквы по этому же квадрату:

Таблица координат								
Координата горизонтальная:	**3**	**2**	**4**	**3**	**4**	**3**	**4**	**5**
Координата вертикальная:	**4**	**5**	**5**	**4**	**3**	**1**	**1**	**4**
Буква:	**S**	**W**	**Y**	**S**	**O**	**C**	**D**	**U**

Таким образом после шифрования получаем:

До шифрования:	**SOMETEXT**
После шифрования:	**SWYSOCDU**

Метод 2

Усложненный вариант, который заключается в следующем: полученный первичный шифротекст (*) шифруется вторично. При этом он выписывается без разбиения на пары:

3425453443314154

Полученная последовательность цифр сдвигается циклически влево на один шаг(нечетное количество шагов):

4254534433141543

Эта последовательность вновь разбивается в группы по два:

42 54 53 44 33 14 15 43

и по таблице заменяется на окончательный шифротекст:

Таблица координат								
Координата горизонтальная:	**4**	**5**	**5**	**4**	**3**	**1**	**1**	**4**
Координата вертикальная:	**2**	**4**	**3**	**4**	**3**	**4**	**5**	**3**
Буква:	**I**	**U**	**P**	**T**	**N**	**Q**	**V**	**O**

Таким образом после шифрования получаем:

Результат	
До шифрования:	**SOMETEXT**
После шифрования:	**IUPTNQVO**

Метод 3

Добавление ключа

На первый взгляд шифр кажется очень нестойким, но для его реальной оценки следует учитывать два фактора:

1. *возможность заполнить квадрат Полибия буквами произвольно, а не только строго по алфавиту;*
1. *возможность периодически заменять квадраты.*

Тогда анализ предыдущих сообщений ничего не дает, так как к моменту раскрытия шифра он может быть заменен.

Буквы могут вписываться в таблицу в произвольном порядке — заполнение таблицы в этом случае и является ключом. Для латинского алфавита в первую клетку можно вписать одну из 25 букв, во вторую — одну из 24, в третью — одну из 23 и т. д. Получаем максимальное количество ключей для шифра на таблице латинского алфавита:

$$N = 25 * 24 * 23 * ... * 2 * 1 = 25!$$

Соответственно для дешифрования сообщения потребуется не только знание алфавита, но и ключа, с помощью которого составлялась таблица шифрования. Но произвольный порядок букв тяжело запомнить, поэтому пользователю шифра необходимо постоянно иметь при себе ключ — квадрат. Появляется опасность тайного ознакомления с ключом посторонних лиц. В качестве компромиссного решения был предложен ключ — пароль. Пароль выписывается без повторов букв в квадрат; в оставшиеся клетки в алфавитном порядке выписываются буквы алфавита, отсутствующие в пароле.

Пример

Зашифруем слово «SOMETEXT», используя ключ «DRAFT». Составим предварительно таблицу шифрования с данным ключом, записывая символы ключа по порядку в таблицу, после них остальной алфавит:

	1	**2**	**3**	**4**	**5**
1	D	R	A	F	T
2	B	C	E	G	H
3	I	K	L	M	N
4	O	P	Q	S	U
5	V	W	X	Y	Z

Преобразуем сообщение в координаты по квадрату Полибия:

Таблица координат								
Буква:	**S**	**O**	**M**	**E**	**T**	**E**	**X**	**T**
Координата горизонтальная:	**4**	**1**	**4**	**3**	**5**	**3**	**3**	**5**
Координата вертикальная:	**4**	**4**	**3**	**2**	**1**	**2**	**5**	**1**

Считаем координаты по строкам:
41 43 53 35 44 32 12 51
Преобразуем координаты в буквы по этому же квадрату

Таблица координат								
Координата горизонтальная:	**4**	**4**	**5**	**3**	**4**	**3**	**1**	**5**
Координата вертикальная:	**1**	**3**	**3**	**5**	**4**	**2**	**2**	**1**
Буква:	**F**	**M**	**N**	**X**	**S**	**E**	**B**	**T**

Таким образом после шифрования получаем:
Результат

До шифрования: **SOMETEXT**

После шифрования: **FMNXSEBT**

Практическая часть

Задание №1

Зашифровать методом двойной перестановки индивидуальную фразу. Попытаться выбрать максимально криптостойкую таблицу.

Задание №2

Зашифровать квадратом Полибия индивидуальную фразу **тремя** способами.

Контрольные вопросы:

1. Дать определение симметричному кроптоалгоритму
2. Опишите метод шифрующих таблиц
3. Опишите метод двойной перестановки
4. Отличия трех методов шифрования в квадрате Полибия
5. Описать принцип шифрования квадрата Полибия

Лабораторная работа № 6. Исследование ассиметричных крипроалгоритмов

Цель работы: изучить и закрепить умение реализации криптосистемы RSA и криптосистемы Эль-Гамаля

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР6_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других.

RSA относится к так называемым *асимметричным алгоритмам*, у которых ключ шифрования не совпадает с ключом расшифровки. Один из ключей доступен всем и называется *открытым ключом*, другой хранится только у хозяина и никому неизвестен. С помощью одного ключа можно производить операции только в одну сторону. Если сообщение зашифровано с помощью одного ключа, то расшифровать его можно только с помощью другого. Имея один из ключей практически невозможно найти другой ключ, если разрядность ключа высока.

Описание RSA

Алгоритм RSA состоит из следующих пунктов:

1. Выбрать простые числа p и q.
2. Вычислить n = p*q.
3. Вычислить m = (p – 1) * (q – 1).
4. Выбрать число d взаимно простое с m.
5. Выбрать число e так, чтобы e*d = 1 (mod m).

Числа e и d являются ключами RSA. Шифруемые данные необходимо разбить на блоки – числа от 0 до n – 1. Шифрование и расшифровка данных производятся следующим образом:

- ✓ шифрование: $b = a^e \pmod n$;
- ✓ расшифрование: $a = b^d \pmod n$.

Следует также отметить, что ключи e и d равноправны, то есть сообщение можно шифровать как ключом e, так и ключом d, при этом расшифровка должна быть произведена с помощью другого ключа.

Весь алгоритм расписан в таблице:

Этап	Описание операции	Результат операции
Генерация ключей	Выбрать два простых различных числа	$p = 3557$, $q = 2579$
	Вычислить модуль (произведение)	$n = p \cdot q = 3557 \cdot 2579 = 9173503$
	Вычислить функцию Эйлера	$\varphi(n) = (p-1)(q-1) = 9167368$
	Выбрать открытую экспоненту	$e = 3$
	Вычислить секретную экспоненту	$d = e^{-1} \mod \varphi(n)$ $d = 6111579$
	Опубликовать открытый ключ	$\{e, n\} = \{3, 9173503\}$
	Сохранить закрытый ключ	$\{d, n\} = \{6111579, 9173503\}$
Шифрование	Выбрать текст для зашифровки	$m = 111111$
	Вычислить шифротекст	$c = E(m) = m^e \mod n = 111111^3 \mod 9173503 = 4051753$
Расшифрование	Вычислить исходное сообщение	$m = D(c) = c^d \mod n = 4051753^{6111579} \mod 9173503 = 111111$

Реализация элементов схемы шифрования Эль-Гамаля

Генерация ключей

1. Генерируется случайное простое число p длины n битов.
2. Выбирается случайный примитивный элемент g.
3. Выбирается случайное целое число x такое, что $1 < x < p-1$.
4. Вычисляется $y = g^x \bmod p$.
5. Открытым ключом является тройка (p, g, y), закрытым ключом - число x.

Шифрование

Сообщение M шифруется следующим образом:

1. Выбирается сессионный ключ — случайное целое число k такое, что $1 < k < p-1$
2. Вычисляются числа $a = g^k \bmod p$ и $b = y^k M \bmod p$.
3. Пара чисел (a, b) является шифротекстом.

Нетрудно видеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

Расшифрование

Зная закрытый ключ x, исходное сообщение можно вычислить из шифротекста (a, b) по формуле:

$M = b(a^x)^{-1} \bmod p.$

При этом нетрудно проверить, что

$(a^x)^{-1} \equiv g^{-kx} \pmod p$

и поэтому

$b(a^x)^{-1} \equiv (y^k M)g^{-xk} \equiv (g^{xk} M)g^{-xk} \equiv M \pmod p.$

Для практических вычислений больше подходит следующая формула:

$M = b(a^x)^{-1} \bmod p = b \cdot a^{(p-1-x)} \bmod p$ (рис. 6.1)

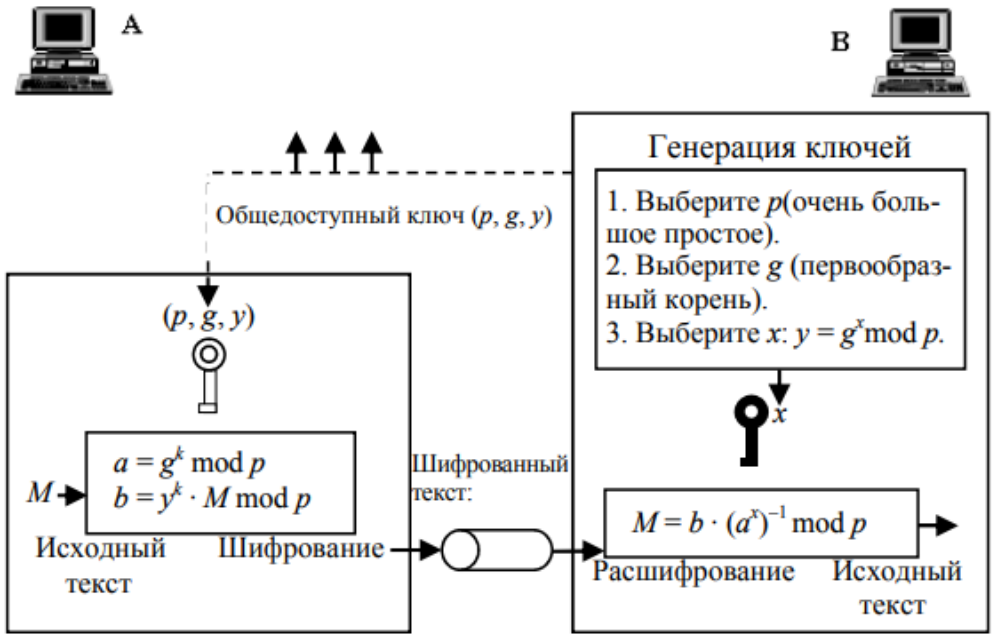


Рис. 6.1

Пример

Шифрование

Допустим что нужно зашифровать сообщение $M = 5.$

Произведем генерацию ключей :

пусть $p = 11, g = 2$. Выберем $x = 8$ - случайное целое число x такое,что $1 < x < p.$

Вычислим $y = g^x \bmod p = 2^8 \bmod 11 = 3.$

Итак , открытым является тройка $(p, g, y) = (11, 2, 3)$,а закрытым ключом является число $x = 8.$

Выбираем случайное целое число k такое, что 1 < k < (p − 1). Пусть $k = 9.$

Вычисляем число $a = g^k \bmod p = 2^9 \bmod 11 = 512 \bmod 11 = 6.$

Вычисляем число $b = y^k M \bmod p = 3^9 5 \bmod 11 = 19683 \cdot 5 \bmod 11 = 9.$

Полученная пара $(a, b) = (6, 9)$ является шифротекстом.

Расшифрование

Необходимо получить сообщение $M = 5$ по известному шифротексту $(a, b) = (6, 9)$ и закрытому ключу $x = 8.$

Вычисляем M по формуле : $M = b(a^x)^{-1} \bmod p = 9(6^8)^{-1} \mod 11 = 5$

Получили исходное сообщение $M = 5.$

Практическая часть

Задание №1

Написать программу которая зашифрует и расшифрует текст введенный с клавиатуры, зашифрует и расшифрует его алгоритмом RSA.

Числа p и q выбрать самим.

В отчете должно сдержаться код программы, полученные результаты.

Код функции вычисления модуля на C++

```
template <typename T>
T modpow(T base, T exp, T modulus) {
        base %= modulus;
        T result = 1;
        while (exp > 0) {
                if (exp & 1) result = (result * base) % modulus;
                base = (base * base) % modulus;
                exp >>= 1;
        }       return result; }
```

Задание №2

Написать программу которая сгенерирует ключи, зашифрует и расшифрует текст введенный с клавиатуры алгоритмом Эль-Гамаля.

Ниже приведен пример программы реализующий данный алгоритм. Напишите на его примере программу обрабатывающую нашу задачу.

Приложение: код рабочей программы на C++

```
int power(int a, int b, int n){// a^b mod n
   int tmp=a;
   int sum=tmp;
   for(int i=1;i<b;i++){
      for(int j=1;j<a;j++){
         sum+=tmp;
         if(sum>=n){
            sum-=n;
         }
      }
      tmp=sum;
   }
   return tmp; }
 int mul(int a, int b, int n){// a*b mod n
   int sum=0;

   for(int i=0;i<b;i++){
      sum+=a;

      if(sum>=n){
         sum-=n;
      }    }
   return sum; }
/*********************************************************
   p - простое число
   0 < g < p-1
   0 < x < p-1
   m - шифруемое сообщение m < p
*********************************************************/
```

```
        void crypt(int p,int g,int x, string inFileName,string outFileName){
           ifstream inf(inFileName.c_str());
           ofstream outf(outFileName.c_str());
            int y=power(g,x,p);
               wcout<<"Открытый ключ (p,g,y)="<<"("<<p<<","<<g<<","<<y<<")"<<endl;
           wcout<<"Закрытый ключ x="<<x<<endl;
               wcout<<"\nШифруемый текст:"<<endl;
            while(inf.good()){
              int m=inf.get();
              if(m>0){
        wcout<<(char)m;
                  int k=rand()%(p-2)+1; // 1 < k < (p-1)
        int a= power(g,k,p);
        int b= mul(power(y,k,p),m,p);
        outf<<a<<" "<<b<<" ";        }     }
   wcout<<endl;
    inf.close();
   outf.close(); }
 void decrypt(int p,int x,string inFileName,string outFileName){
   ifstream inf(inFileName.c_str());
   ofstream outf(outFileName.c_str());
   wcout<<"\nДешифрованый текст:"<<endl;
    while(inf.good()){
      int a=0;
      int b=0;
      inf>>a;
      inf>>b;
            if(a!=0&&b!=0){
        //wcout<<a<<" "<<b<<endl;
               int deM=mul(b,power(a,p-1-x,p),p);// m=b*(a^x)^(-1)mod p =b*a^(p-1-x)mod p -
трудно было  найти нормальную формулу, в ней вся загвоздка
        char m=static_cast<char>(deM);
        outf<<m;
        wcout<<m;
      }     }
       wcout<<endl;
    inf.close();
   outf.close();
}
int main(){
   srand(time(NULL));
   crypt(593,123,8, "in.txt","out_crypt.txt");
   decrypt(593,8,"out_crypt.txt","out_decrypt.txt");
      return 0; }
```

Контрольные вопросы:

1. Дать определение понятию «Ассиметричный криптоалгоритм»
2. Описать принцип крипросистемы RSA
3. Описать принцип крипросистемы Эль-Гамаля
4. Описать зашифровку текста алгоритмом Эль-Гамаля
5. Описать расшифровку текста алгоритмом Эль-Гамаля

Лабораторная работа № 7. Исследование криптографических хэш-функций

Цель работы: реализовать криптосистему Дифи-Хеллмана и подсчета контрольных сумм на примере 128-битного алгоритма хеширования md5

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР7_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Реализация криптосистемы Дифи-Хеллмана

Генерация ключей

В 1976 году после публичной критики алгоритма DES и указания на сложность обработки секретных ключей Уитфилд Диффи (Whitfield Diffie) и Мартин Хеллман (Martin Hellman) опубликовали свой алгоритм обмена ключами. Это была первая публикация на тему криптографии с открытым ключом и, возможно, самый большой шаг вперед в области криптографии, сделанный когда-либо.

Из-за невысокого быстродействия, свойственного асимметричным алгоритмам, алгоритм Диффи-Хеллмана не предназначен для шифрования данных. Он был ориентирован на передачу секретных ключей DES, ARS или других подобных алгоритмов через небезопасную среду. В большинстве случаев алгоритм Диффи-Хеллмана не используется для шифрования сообщений, потому что он, в зависимости от реализации, от 10 до 1000 раз медленнее алгоритма DES.

До алгоритма Диффи-Хеллмана было сложно совместно использовать зашифрованные данные из-за проблем хранения ключей и передачи информации. Подробнее об этом будет еще сказано. В большинстве случаев передача информации по каналам связи небезопасна, потому что сообщение может пройти десятки систем, прежде чем оно достигнет потенциального адресата, и нет никаких гарантий, что по пути никто не сможет взломать секретный ключ. Уитфилд Диффи и Мартин Хеллман предложили зашифровывать секретный ключ DES по алгоритму Диффи-Хеллмана на передающей стороне и пересылать его вместе с сообщением, зашифрованным с использованием DES. Тогда на другом конце его сможет расшифровать только получатель сообщения.

На практике **обмен ключами** по алгоритму Диффи-Хеллмана происходит по следующей схеме.

1. Два участника обмена договариваются о двух числах. Один выбирает большое простое число, а другой – целое число, меньшее числа первого участника. Переговоры они могут вести открыто, и это никак не отразится на безопасности.

2. Каждый из двух участников, независимо друг от друга, генерирует другое число, которое они будут хранить в тайне. Эти числа выполняют роль секретного ключа. Далее в вычислениях используются секретный ключ и два предыдущих целых числа. Результат вычислений посылается участнику обмена, и он играет роль открытого ключа.

3. Участники обмена обмениваются открытыми ключами. Далее они, используя собственный секретный ключ и открытый ключ партнера, конфиденциально вычисляют ключ сессии. Каждый партер вычисляет один и тот же ключ сессии.

4. Ключ сессии может использоваться как секретный ключ для другого алгоритма шифрования, например DES. Никакое третье лицо, контролирующее обмен, не сможет вычислить ключ сессии, не зная один из секретных ключей.

Самое сложное в алгоритме Диффи-Хеллмана обмена ключами – это понять, что в нем фактически два различных независимых цикла шифрования. Алгоритм Диффи-Хеллмана применяется для обработки небольших сообщений от отправителя получателю. Но в этом маленьком сообщении передается секретный ключ для расшифровки большого сообщения.

Сильная сторона алгоритма - никто не сможет скомпрометировать секретное сообщение, зная один или даже два открытых ключа получателя и отправителя. В качестве секретных и открытых ключей используются очень большие целые числа. Алгоритм Диффи-Хеллмана основан на полезных для криптографии свойствах дискретных логарифмов.

Пример

Ева – криптоаналитик. Она читает пересылку Боба и Алисы, но не изменяет содержимого их сообщений.

- s = секретный ключ. s = 2
- g = простое число меньшее p. g = 5
- p = открытое простое число. p = 23
- a = секретный ключ Алисы. a = 6
- A = открытый ключ Алисы. $A = g^a \bmod p = 8$
- b = секретный ключ Боба. b = 15
- B = открытый ключ Боба. $B = g^b \bmod p = 19$

Alice	
Знает	Не знает
p = 23	**b = ?**
g = 5	
a = 6	
$A = 5^6 \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^6 \bmod 23 = 2$	
$s = 8^b \bmod 23 = 2$	
$s = 19^6 \bmod 23 = 8^b \bmod 23$	
s = 2	

Bob	
Знает	Не знает
p = 23	**a = ?**
g = 5	
b = 15	
$B = 5^{15} \bmod 23 = 19$	
$A = 5^a \bmod 23 = 8$	
$s = 8^{15} \bmod 23 = 2$	
$s = 19^a \bmod 23 = 2$	
$s = 8^{15} \bmod 23 = 19^a \bmod 23$	
s = 2	

Eve	
Знает	Не знает
p = 23	**a = ?**
g = 5	**b = ?**
	s = ?
$A = 5^a \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^a \bmod 23$	
$s = 8^b \bmod 23$	
$s = 19^a \bmod 23 = 8^b \bmod 23$	

Реализация элементов подсчета контрольных сумм методом md5

Основные понятия

Хеширование (иногда хэширование, англ. hashing) - преобразование входного массива данных произвольной длины в выходную строку фиксированной длины. Такие преобразования также называются хеш-функциями или функциями свёртки, входной массив – прообразом, а результаты преобразования - хешем, хеш-кодом, хеш-образом, цифровым отпечатком или дайджестом сообщения (англ. message digest).

Хеш-функция – легко вычислимая функция, преобразующая исходное сообщения произвольной длины (прообраз) в сообщение фиксированное длины (хеш-образ), для которой не существует эффективного алгоритма поиска коллизий.

Коллизией для функции h называется пара значений x, y, $x \neq y$, такая, что h(x) = h(y). Т.о. хеш-функция должна обладать следующими свойствами:

- для данного значения h(x) невозможно найти значение аргумента x. Такие хеш-функции называют стойкими в смысле обращения или стойкими в сильном смысле;

- для данного аргумента x невозможно найти другой аргумент y такой, что h(x) = h(y). Такие хеш-функции называют стойкими в смысле вычисления коллизий или стойкими в слабом смысле.

В случае, когда значение хеш-функции зависит не только от прообраза, но и закрытого ключа, то это значение называют кодом проверки подлинности сообщений (Message

Authentication Code, MAC), кодом проверки подлинности данных (Data Authentication Code, DAC) или имитовставкой.

На практике хеш-функции используют в следующих целях:

- для ускорения поиска данных в БД;
- для проверки целостности и подлинности сообщений;
- для создания сжатого образа, применяемого в процедурах ЭЦП;
- для защиты пароля в процедурах аутентификации.

Ускорения поиска данных. Например, при записи текстовых полей в базе данных может рассчитываться их хеш-код и данные могут помещаться в раздел, соответствующий этому хеш-коду. Тогда при поиске данных надо будет сначала вычислить хеш-код текста и сразу станет известно, в каком разделе их надо искать, т.е. искать надо будет не по всей базе, а только по одному её разделу (это сильно ускоряет поиск).

Бытовым аналогом хеширования в данном случае может служить помещение слов в словаре по алфавиту. Первая буква слова является его хеш-кодом, и при поиске мы просматриваем не весь словарь, а только раздел с нужной буквой.

Процедура вычисления (стандартная схема алгоритма) хеш-функции представлена на следующем рисунке.

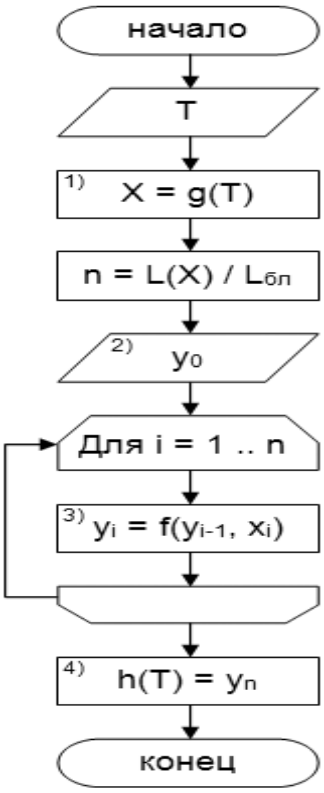


Рис.1. Процедура вычисления значения хеш-функции

(1) К исходному сообщению T добавляется вспомогательная информация (например, длина прообраза, вспомогательные символы и т.д.) так, чтобы длина прообраза X стала кратной величине Lбл, определенной спецификацией (стандартом) хеш-функции.

(2) Для инициализации процедуры хеширования используется синхропосылка y0.

(3) Прообраз X разбивается на n блоков xi (i = 1 .. n) фиксированной длины Lбл, над которыми выполняется однотипная процедура хеширования f(yi-1, xi), зависящая от результата хеширования предыдущего блока yi-1.

(4) Хеш-образом h(T) исходного сообщения T будет результат процедуры хеширования yn, полученный после обработки последнего блока xn.

MD5

MD5 (англ. Message Digest 5) – 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института (Massachusetts Institute of Technology, MIT) в 1991 году. Является улучшенной в плане безопасности версией MD4.

Ниже приведен алгоритм вычисления хеша.

1. Выравнивание потока.

В конец исходного сообщения, длиной L, дописывают единичный бит, затем необходимое число нулевых бит так, чтобы новый размер L' был сравним с 448 по модулю 512 (L’ mod 512 = 448). Добавление нулевых бит выполняется, даже если новая длина, включая единичный бит, уже сравнима с 448.

2. Добавление длины сообщения.

К модифицированному сообщению дописывают 64-битное представление длины данных (количество бит в сообщении). Т.е. длина сообщения T становится кратной 512 (T mod 512 = 0). Если длина исходного сообщения превосходит 264 - 1, то дописывают только младшие 64 бита. Кроме этого, для указанного 64-битного представления длины вначале записываются младшие 32 бита, а затем старшие 32 бита.

3. Инициализация буфера.

Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения (шестнадцатеричное представление):

A = 67 45 23 01;

B = EF CD AB 89;

C = 98 BA DC FE;

D = 10 32 54 76.

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором.

4. Вычисление хеша в цикле.

Исходное сообщение разбивается на блоки T, длиной 512 бит. Для каждого блока в цикле выполняется процедура, приведенная на рис.9.2. Результат обработки всех блоков исходного сообщения в виде объединения 32-битных значений переменных ABCD и будет являться хешем.

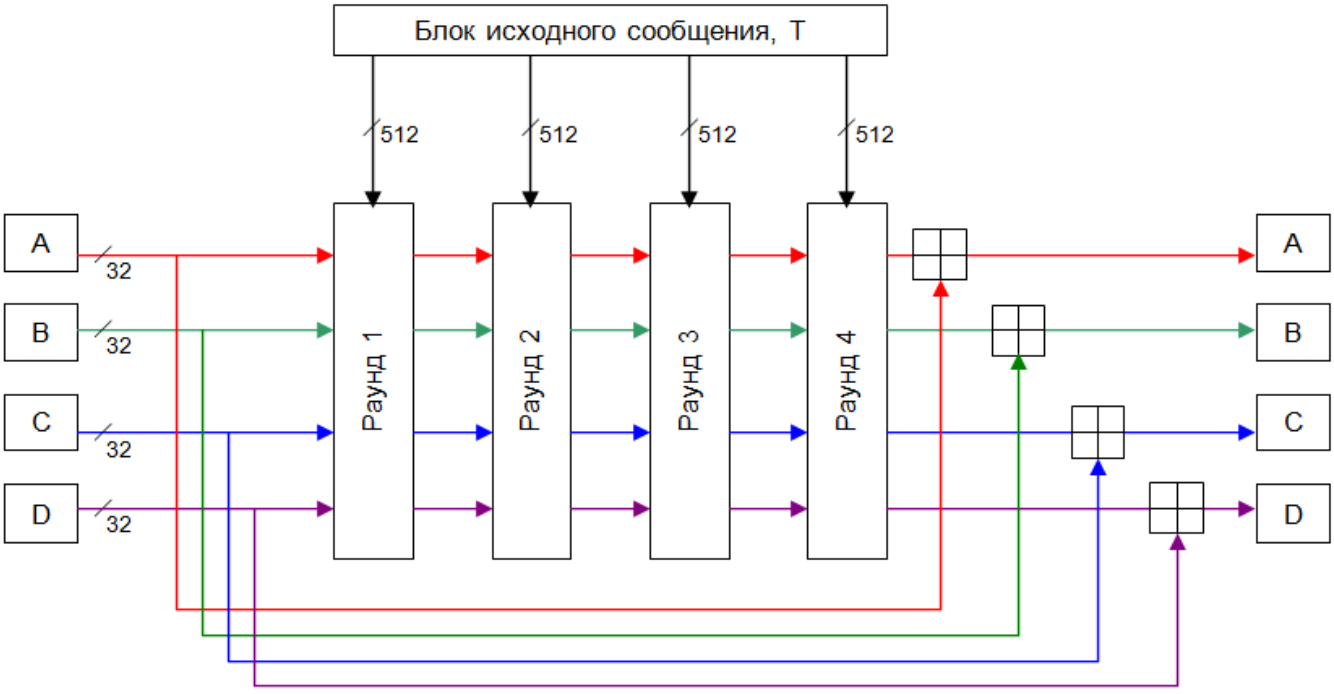


Рис.2. Шаг основного цикла вычисления хеша

В каждом раунде над переменными ABCD и блоком исходного текста T в цикле (16 итераций) выполняются однотипные преобразования по следующей схеме.

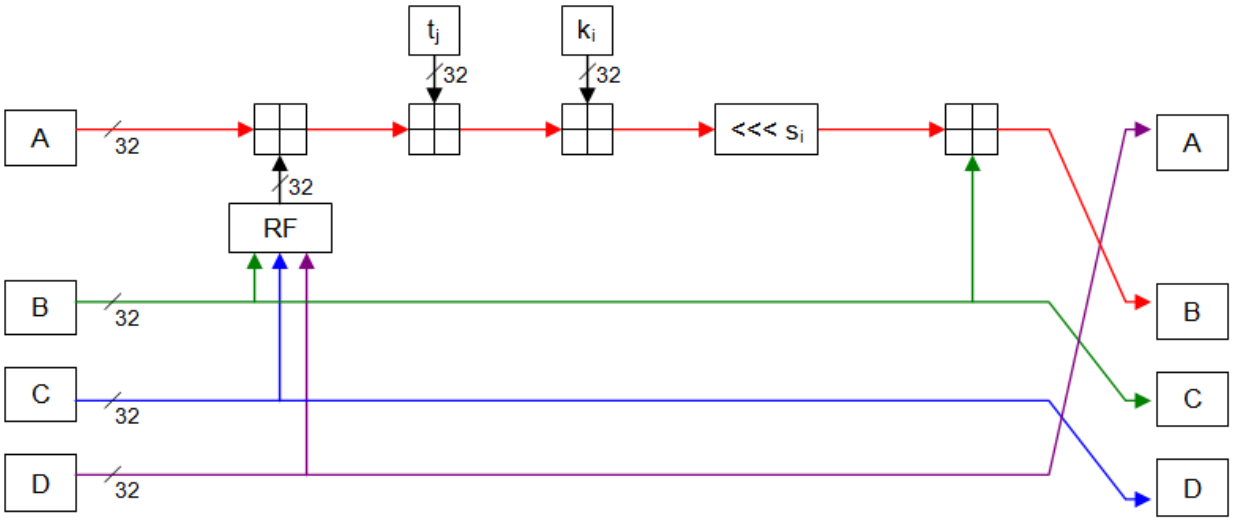


Рис.3. Одна итерация цикла раунда

Условные обозначения.

1) RF - раундовая функция, определяемая по следующей таблице.

Таблица 1. Раундовые функции RF

№ раунда	Обозначение функции	Формула расчета
1	F	$F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
2	G	$G(B, C, D) = (B \wedge D) \vee (\neg D \wedge C)$
3	H	$H(B, C, D) = B \oplus C \oplus D$
4	I	$I(B, C, D) = C \oplus (\neg D \vee B)$

2) tj - j-ая 32-битовая часть блока исходного сообщения T с обратным порядком следования байт;

3) ki - целая часть константы, определяемой по формуле

$ki = 232 * | \sin(i + 16 * (r - 1)) |,$ (9.1)

где i – номер итерации цикла (i = 1..16);

r – номер раунда (r = 1..4).

Аргумент функции sin измеряется в радианах.

4) <<< si – циклический сдвиг влево на si разрядов.

Используемая 32-битовая часть блока исходного сообщения tj и величина циклического сдвига влево si зависят от номера итерации и приведены в следующей таблице.

Таблица 2. Величины, используемые на шаге цикла раунда

№ итерации		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Раунд 1	t_j	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}
	s_i	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
Раунд 2	t_j	t_2	t_7	t_{12}	t_1	t_6	t_{11}	t_{16}	t_5	t_{10}	t_{15}	t_4	t_9	t_{14}	t_3	t_8	t_{13}
	s_i	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
Раунд 3	t_j	t_6	t_9	t_{12}	t_{15}	t_2	t_5	t_8	t_{11}	t_{14}	t_1	t_4	t_7	t_{10}	t_{13}	t_{16}	t_3
	s_i	4	11	16	23	4	11	16	23	4	11	16	23	4	11	16	23
Раунд 4	t_j	t_1	t_8	t_{15}	t_6	t_{13}	t_4	t_{11}	t_2	t_9	t_{16}	t_7	t_{14}	t_5	t_{12}	t_3	t_{10}
	s_i	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

После 4 раундов новое (модифицированное) значение каждой из переменных ABCD складывается (⊞) с исходным (значением переменной до 1-го раунда).

4. Перестановка байт в переменных ABCD. После обработки всех блоков исходного сообщения для каждой переменной выполняется обратная перестановка байт.

Практическая часть

Задание №1

Написать программу которая получит общий секретный ключ из доступных публичных ключей с помощью алгоритма Диффи-Хеллмана. Сгенерируйте секретный ключ Алисы и Боба, и высчитайте секретный ключ, удостоверьтесь что его одинаково смогут получить и Алиса и Боб.

Код алгоритма высчитывания модуля от степенной функции

```
int power(int a, int b, int n){// a^b mod n
   int tmp=a;
   int sum=tmp;
   for(int i=1;i<b;i++){
      for(int j=1;j<a;j++){
         sum+=tmp;
         if(sum>=n){
            sum-=n;
         }       }
      tmp=sum;
   }     return tmp; }
```

Задание №2

Разобраться в приведенной программе.

Посчитать контрольную сумму своего ФИО. А также ФИО где все буквы прописные и все строчные. Сравнить результаты

Рабочий код подсчета сумм MD5

```
#include<string>
#include<iostream>
using namespace std;
typedef unsigned int uint;
string to_hex(uint value);
uint F(uint X, uint Y, uint Z);
uint G(uint X, uint Y, uint Z);
uint H(uint X, uint Y, uint Z);
uint I(uint X, uint Y, uint Z);
uint rotate_left(uint value, int shift);
string get_md5(string in);

int _tmain(int argc, _TCHAR* argv[])
{
	cout << get_md5("") << endl;
	return 0;
}
uint F(uint X, uint Y, uint Z) { return ((X & Y) | ((~X) & Z)); }
uint G(uint X, uint Y, uint Z) { return (X & Z) | (Y & (~Z)); }
uint H(uint X, uint Y, uint Z) { return X ^ Y ^ Z; }
uint I(uint X, uint Y, uint Z) { return Y ^ (X | (~Z)); }
uint rotate_left(uint value, int shift) { return value << shift | value >> (32 - shift); }
string to_hex(uint value)
{
	string out;
	unsigned char hex;
	char hex_res[3];
	while (value)
	{
		hex = value % 256;
		_itoa_s(hex, hex_res, 16);
		if (hex_res[1] == '\0')
		{
			hex_res[1] = hex_res[0];
			hex_res[0] = '0';
			hex_res[2] = '\0';
		}
		out.append(hex_res);
		value /= 256;
```

```
	}
	return out;
}
string get_md5(string in)
{
	int length = in.length();; //получаем длину входного сообщения.
	int rest = length % 64; //остаток от деления на 64байта.
	int size = 0; //тут будет храниться размер сообщения после первых 2ух шагов.

	//Шаг 1.
	if (rest < 56) //если остаток от деления на 64 меньше 56
		size = length - rest + 56 + 8; //подгоняем размер так, что бы он был кратен 64(+8 байт для 2ого шага).
	else //иначе (если остаток больше 56)
		size = length + 64 - rest + 56 + 8; //подгоняем размер так, что бы он был кратен 64(+8 байт для 2ого шага).

	unsigned char *msg_for_decode = new unsigned char[size]; //создаем динамический массив для хранения сообщения, которое далее будет кодироваться

	for (int i = 0; i < length; i++) //первые length элементов cIn
		msg_for_decode[i] = in[i]; //заполняем символами входного сообщения

	msg_for_decode[length] = 0x80; //ставим в конец сообщения единичный бит.

	for (int i = length + 1; i < size; i++) //а все остальное
		msg_for_decode[i] = 0; //заполняем нулями

	//Шаг 2.
	__int64 bit_length = (uint)(length)* 8; //длина сообщения в битах.

	for (int i = 0; i < 8; i++) //последние 8 байт
		msg_for_decode[size - 8 + i] = (unsigned char)(bit_length >> i * 8); //заполняем 64-битным представлением длины данных до выравнивания

	//Шаг 3.
	uint A = 0x67452301, B = 0xefcdab89, C = 0x98badcfe, D = 0x10325476; //Инициализируем начальные значения регистров.
	uint T[64]; //64-элементная таблица данных (констант).

	for (int i = 0; i<64; i++) //всю таблицу констант
		T[i] = uint(pow(2, 32)*fabs(sin(i + 1)));; //заполняем в соответствии с алгоритмом.

	//объявляем массив X, в котором будет 32-разрядное представление сообщения.
	uint *X = (uint*)(msg_for_decode); //загоняем в массив X сообщение msg_for_decode.

	//Шаг 4.
	uint AA, BB, CC, DD;

	for (int i = 0; i<size / 4; i += 16)
	{
		AA = A; BB = B; CC = C; DD = D;

		//раунд 1
		A = B + rotate_left((A + F(B, C, D) + X[i + 0] + T[0]), 7);
		D = A + rotate_left((D + F(A, B, C) + X[i + 1] + T[1]), 12);
		C = D + rotate_left((C + F(D, A, B) + X[i + 2] + T[2]), 17);
		B = C + rotate_left((B + F(C, D, A) + X[i + 3] + T[3]), 22);

		A = B + rotate_left((A + F(B, C, D) + X[i + 4] + T[4]), 7);
		D = A + rotate_left((D + F(A, B, C) + X[i + 5] + T[5]), 12);
		C = D + rotate_left((C + F(D, A, B) + X[i + 6] + T[6]), 17);
		B = C + rotate_left((B + F(C, D, A) + X[i + 7] + T[7]), 22);

		A = B + rotate_left((A + F(B, C, D) + X[i + 8] + T[8]), 7);
```

```
D = A + rotate_left((D + F(A, B, C) + X[i + 9] + T[9]), 12);
C = D + rotate_left((C + F(D, A, B) + X[i + 10] + T[10]), 17);
B = C + rotate_left((B + F(C, D, A) + X[i + 11] + T[11]), 22);

A = B + rotate_left((A + F(B, C, D) + X[i + 12] + T[12]), 7);
D = A + rotate_left((D + F(A, B, C) + X[i + 13] + T[13]), 12);
C = D + rotate_left((C + F(D, A, B) + X[i + 14] + T[14]), 17);
B = C + rotate_left((B + F(C, D, A) + X[i + 15] + T[15]), 22);

//раунд 2
A = B + rotate_left((A + G(B, C, D) + X[i + 1] + T[16]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 6] + T[17]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 11] + T[18]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 0] + T[19]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 5] + T[20]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 10] + T[21]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 15] + T[22]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 4] + T[23]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 9] + T[24]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 14] + T[25]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 3] + T[26]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 8] + T[27]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 13] + T[28]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 2] + T[29]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 7] + T[30]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 12] + T[31]), 20);

//раунд 3
A = B + rotate_left((A + H(B, C, D) + X[i + 5] + T[32]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 8] + T[33]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 11] + T[34]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 14] + T[35]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 1] + T[36]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 4] + T[37]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 7] + T[38]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 10] + T[39]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 13] + T[40]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 0] + T[41]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 3] + T[42]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 6] + T[43]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 9] + T[44]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 12] + T[45]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 15] + T[46]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 2] + T[47]), 23);

//раунд 4
A = B + rotate_left((A + I(B, C, D) + X[i + 0] + T[48]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 7] + T[49]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 14] + T[50]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 5] + T[51]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 12] + T[52]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 3] + T[53]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 10] + T[54]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 1] + T[55]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 8] + T[56]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 15] + T[57]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 6] + T[58]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 13] + T[59]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 4] + T[60]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 11] + T[61]), 10);
```

```
            C = D + rotate_left((C + I(D, A, B) + X[i + 2] + T[62]), 15);
            B = C + rotate_left((B + I(C, D, A) + X[i + 9] + T[63]), 21);

            A += AA;
            B += BB;
            C += CC;
            D += DD;
        }

        delete[]msg_for_decode; //не забываем освободить память =)
        //Шаг 5.
        string res = to_hex(A) + to_hex(B) + to_hex(C) + to_hex(D); //заполняем выходную строку hex-//представлением, полученных в шаге 4, регистров.

        return res; //возвращаем строку с хеш-кодом.
}
```

Контрольные вопросы

1. Как происходит обмен ключами по алгоритму Дифи-Халлмана?
2. Самое сложное в алгоритме Дифи-Халлмана.
3. Сильная сторона алгоритма Дифи-Халлмана
4. Дайте определение понятиям: «хеширование», «хеш-функция», «коллизия».
5. В каких целях используют хеш-функции на практике?
6. Приведите стандартную схему алгоритма генерации хеш-образа.
7. Как называется хеш-образ, полученный с применением закрытого ключа шифрования?

Лабораторная работа № 8. Защита от компьютерных вирусов

Цель занятия: закрепить сведения о классификации компьютерных вирусов, способах их распространения, способах борьбы с ними; закрепить сведения о классификации и назначении антивирусных программ.

Порядок выполнения работы:

1. Ответить на контрольные вопросы
2. Запишите результаты выполнения пункта 7.
3. Запишите информацию из пункта 8 выполнения работы.
4. Запишите информацию из пункта 10 выполнения задания: о чём может предупреждать программа пользователя.
5. Запишите информацию из пункта 11 выполнения задания.

Отчет переименовать следующим образом: **ЗКИ_ЛР8_Фамилия_№группы** и сдать на проверку преподавателю

Теоретическая часть

Наиболее защищенный компьютер — это тот компьютер, который отключен от сети и заперт в сейф.

Понятие вируса.

Официальное появление *первого компьютерного вируса* датируется 1981 годом, задолго до выхода первой версии Microsoft Windows. Этот вирус, замаскированный под компьютерную игру, атаковал наиболее популярный компьютер того времени — Apple II. Распространялся он с черепашьей скоростью (с помощью дискет).

Согласно подсчетам экспертов, объем *malware* (общепринятое название всех видов вредоносных программ) возрастает более чем на 15 % в год. Согласно данным компании Sophos, разработчика антивирусных программ, каждый день появляются примерно 30 новых вирусов, а перечень активных вирусов пополняется 10 тыс. новых наименований в год.

Вирус — это часть программного кода, которая тиражируется путем добавления в другой объект, обычно незаметно и без разрешения пользователя.

Встреча компьютера с вирусом влечет несколько последствий.

- Появление необычных системных сообщений.
- Исчезновение файлов или увеличение их размеров.
- Замедление работы системы.
- Внезапный недостаток дискового пространства.
- Диск становится недоступным.

Классификация вирусов.

Вирусы могут быть безвредными, малоопасными и разрушительными.

Вирусы могут заражать программные файлы, документы (так называемые *макровирусы*) или файловые и дисковые структуры низкого уровня, такие как загрузочный сектор или таблица размещения файлов *(Boot – вирусы)*. *Файловые вирусы* заражают исполнимые файлы, имплантируя в них опасный код. Вирусы могут активизироваться при запуске инфицированной программы; также они могут постоянно находиться в памяти и заражать открываемые пользователем файлы или создавать свои собственные. Когда вирус проникает в компьютер, на котором установлена система Windows, он может изменять значения в системном реестре, замещать собой системные файлы и внедряться в почтовую программу с целью дальнейшего размножения (черви). *Сетевые вирусы* обитают в оперативной памяти компьютеров и не копируют себя на носители данных. Они обитают в сети, когда хотя бы один компьютер включен, поэтому не опасны для индивидуального пользователя. Вирус не обязательно представляет собой отдельную программу и не всегда является деструктивным по своей сути, все зависит от его конкретной разновидности. Хотя основную угрозу для пользователей представляют именно компьютерные вирусы, существует несколько видов вредоносных программ:

Троянский конь представляет собой компьютерную программу, которая маскируется или скрывается в части программы. Некоторые формы троянских коней могут быть запрограммированы на саморазрушение и не оставляют никаких следов, кроме причиненных ими

разрушений. Некоторые хакеры используют троянских коней для получения паролей и отсылки их обратно хакеру. Кроме того, они могут использоваться для банковских мошенничеств, когда небольшие суммы денег снимаются с законных счетов и передаются на секретный счет.

Черви представляют собой программы, которые разрушают компьютерную систему. Они могут проникать в программы обработки данных и подменять или разрушать данные. Как вирусы, они могут причинять большие разрушения, если их не обнаружить вовремя. Намного проще ликвидировать червя или троянского коня, если существует только единственная копия программы-разрушителя.

Логические бомбы подобны программам, используемым для троянских коней. Однако логические бомбы имеют таймер, который взрывает их в заданную дату и время. Например, вирус Michelangelo имеет триггер, установленный на день рождения знаменитого художника Микеланджело – 6 марта. Логические бомбы часто используются недовольными служащими, которые могут установить их на активацию после того, как они оставят компанию. Например, логическая бомба может «взорваться», когда имя этого служащего исключается из платежной ведомости. Благодаря встроенному механизму задержки, логические бомбы активно используются для шантажа. Например, шантажист может послать сообщение, говорящее, что если ему будет выплачена определенная сумма денег, он предоставит инструкцию для отключения логической бомбы.

Смешанные коды представляют собой новый класс изощренных вредоносных программ, которые сочетают в себе характеристики вирусов, червей и 4Зое4Зия43, что позволяет злоумышленнику осуществить особо эффективную атаку. В отличие от большинства доморощенных вирусов, которые распространяются благодаря взлому адресных книг на компьютерах под управлением Windows, целью таких программ являются web-серверы и сети, что значительно повышает их опасность.

Пути проникновения вирусов в компьютер.

Вирусы попадают в вашу компьютерную систему из множества разнообразных *источников* – исполняемых программ, программ и файлов, передаваемых вам, или программного обеспечения, приобретаемого в архивированной форме.

Гибкие диски и компакт-диски могут хранить файлы данных, программ и программное обеспечение операционных систем. Гибкий диск состоит из загрузочного сектора и данных. При необходимости, в загрузочном секторе может храниться информация, нужная для загрузки компьютера. Кроме того, здесь же хранится информация о разделах, информация по управлению загрузкой и информация о размещении файлов. Данные представляют собой всю ту содержательную информацию, которая храниться на гибком диске. Очень легко распространяются вирусы с флеш-карт.

Излюбленным местом обитания вирусов являются загрузочные сектора и исполняемые файлы, хранимые на гибком диске. Помещенные в загрузочном секторе, вирусы могут запускаться при загрузке системы с дискеты. Вирусы, помещенные в исполняемые файлы, запускаются вместе с зараженной программой, после чего начинают свою деятельность.

Если в *локальной сети* заражён хотя бы один компьютер, то вирус моментально распространится и на все остальные компьютеры.

Интернет предоставил пользователям новые возможности, которые увеличивают потенциальную опасность прорех в системе защиты от вирусов.

Места обитания вирусов.

Место обитания вируса связано с его функционированием самым непосредственным образом (как и у настоящих живых вирусов). Вирусные атаки можно даже классифицировать по месту их расположения в компьютере. Типы вирусных атак: *атака загрузочного сектора; инфицирование файла; атака с использованием макросов.*

Вирусы загрузочного сектора инфицируют загрузочный сектор или главную загрузочную запись компьютерной системы. Когда компьютер загружается, вирусная программа активируется. Вирусы загрузочного сектора прежде всего перемещают в другое место записывают исходный загрузочный код и замещают его инфицированным загрузочным кодом. Информация исходного загрузочного сектора переносится на другой сектор диска, который помечается как дефектная область диска и далее не используется.

Поскольку загрузочный сектор – первый элемент, загружаемый при запуске компьютера, обнаружение вирусов загрузочного сектора может оказаться нелегкой задачей. Вирусы загрузочного сектора – один из самых популярных типов вирусов. Они могут распространяться путем использования инфицированных гибких дисков при загрузке компьютера. Это может легко произойти, если при перезагрузке компьютера гибкий диск вставлен в дисковод.

Вирусы, инфицирующие файлы, поражают *исполняемые файлы*. Они могут активироваться только при исполнении файла. Чаще прочих поражаются файлы типов COM, EXE, DLL, BIN, SYS и VXD. Вирусы, инфицирующие файлы, могут становиться резидентными и присоединяться к другим исполняемым программам. Вирусы, инфицирующие файлы, обычно заменяют инструкции загрузки программы исполняемого файла собственными инструкциями. Затем они переносят исходную инструкцию загрузки программы в другой раздел файла. Этот процесс увеличивает размер файла, что может помочь обнаружению вируса.

Вирусы в основе которых лежат макросы (*макровирусы),* исполняют непредусмотренные действия путем использования макроязыка приложения для своего распространения документы. Они могут, например, инфицировать файлы .DOT и .DOC приложения Microsoft Word, а также файлы Microsoft Excel. Эти вирусы относятся к межплатформенным вирусам и могут инфицировать как системы Macintosh, так и PC.

Прочие вирусы могут иметь черты одного или нескольких описанных выше типов.

Вирусы-невидимки (жаргонное название – «стелс-вирусы») при работе пытаются вся как от операционной системы, так и антивирусных программ. Чтобы перехватить все попытки использования операционной системы, вирус должен находиться в памяти. Вирусы невидимки могут скрывать все изменения, которые они вносят в размеры файлов, структуру каталогов или иные разделы операционной системы. Это значительно затрудняет их обнаружение. Чтобы блокировать вирусы-невидимки, их следует обнаружить, когда они находятся в памяти.

Зашифрованные вирусы во время работы шифруют свой вирусный код, что позволяет им предотвратить обнаружение и распознание вируса.

Полиморфные вирусы могут изменять свой внешний вид при каждом инфицировании. Для изменения внешнего вида и затруднения обнаружения они используют механизмы мутаций. Полиморфные вирусы способны принимать более двух миллиардов различных форм, поскольку при каждом инфицировании изменяют алгоритм шифрование.

Многокомпонентные вирусы инфицируют как загрузочные секторы, так и исполняемые файлы. Это один из самых сложных для обнаружения вирусов, поскольку многокомпонентные вирусы могут сочетать некоторые или все методы скрытия своей деятельности, присущие вирусам-невидимкам и полиморфным вирусам.

Самообновляющиеся вирусы, которые появились в самое последнее время, способные скрытно обновляться через Интернет во время сеансов связи.

Проблемы.

Новые вирусы. Сигнатуры новых вирусов появляются постоянно. Когда разрабатывается новый вирус, разработчики антивирусных программ должны «разобрать» его на составные части, проанализировать поведение, добавить его сигнатуру в базу данных антивируса и опубликовать данное обновление. Даже если ваша антивирусная программа настроена на регулярное обновление, какой-то короткий период времени вы не защищены от новейших вирусов. Эта проблема может показаться не столь серьезной в момент начала распространения вируса.

Поскольку новые вирусы появляются непрерывно, никогда не стоит рассчитывать только на антивирусную программу. Для создания нескольких уровней защиты необходимо блокировать исполняемые почтовые вложения и установить все необходимые обновления безопасности.

Ложные тревоги. Иногда антивирусный сканер может принять обычный файл за инфицированный, если база данных антивируса содержит некорректное описание вирусной программы или если алгоритм эвристического анализатора сканера содержит ошибки.

Действия антивирусных программ.

Антивирусная программа должна выполнять *три основные задачи*: обнаружение вируса, удаление вируса, превентивная защита.

Чтобы предотвратить вирусную атаку, антивирусная программа реализует *множество различных методов* обнаружения. Различные антивирусные программы используют некоторые или все методы из следующей группы.

Сканирование цифровой сигнатуры используется для идентификации уникального цифрового кода вируса. Цифровая сигнатура представляет собой предварительно установленный шестнадцатеричный код, наличие которого в файле свидетельствует о его заражении вирусом. Сканирование цифровой сигнатуры представляет собой в высшей степени успешный метод идентификации вирусов. Он, однако, всецело зависит от поддержки базы данных с цифровыми сигнатурами вирусов и тонкостей механизма сканирования. Возможно ложное обнаружение вируса в неповрежденном файле.

Эвристический анализ (или сканирование по заданным правилам) выполняется быстрее, чем сканирование большинством традиционных методов. Этот метод использует набор правил для эффективного анализа файлов и быстро обнаруживает подозрительный вирусный код. Как отмечено в [9], все эвристические методы в той или иной форме выполняют эмулирование исполнения кода вируса. Поэтому, при наличии некоторого опыта, разработчик вируса может защитить свое «изделие» от обнаружения эвристическим анализом. Эвристический анализ склонен к ложным тревогам, и, к сожалению, зависит от корректности набора правил выявления вируса, которые все время изменяются.

Исследование памяти — еще один метод, обычно успешно применяемый для обнару-45ое45ия вирусов. Он зависит от распознания местоположения известных вирусов и их кодов, когда они находятся в памяти. И хотя исследование памяти обычно приводит к успеху, использование такого метода может потребовать значительных ресурсов компьютера. Кроме того, он может вмешиваться в нормальный ход выполнения операций компьютера.

Мониторинг прерываний работает путем локализации и предотвращения вирусных атак, использующих вызовы прерываний. Вызовы прерываний представляют собой запросы различных функций через системные прерывания. Мониторинг прерываний, подобно исследованию памяти, также может отвлечь значительные системные ресурсы. Он может стать причиной проблем при легальных системных вызовах и замедлить работу системы. Из-за большого числа вирусов и легальных системных вызовов, мониторинг прерываний может испытывать трудности в локализации вирусов.

Контроль целостности (известный также как *вычисление контрольных сумм)* просматривает характеристики файлов программ и определяет, были ли они модифицированы вирусным кодом. Этот метод не нуждается в обновлении программного обеспечения, поскольку не зависит от цифровых подписей вирусов. Однако он требует от вас поддержания базы данных контрольных сумм файлов, свободных от вирусов. Контроль целостности не способен обнаруживать пассивные и активные вирусы-невидимки. Кроме того, он не может идентифицировать обнаруженные вирусы по именам или типам.

Непрерывной контроль может быть неподходящим средством для домашнего использования, поскольку может привести к обработке слишком большого объема информации, а это замедляет работу компьютера. На клиентской машине предпочтительнее конфигурировать антивирусную программу на запуск в определенное время. Например, она может запускаться при загрузке компьюте45о или считывании нового файла с гибкого диска. В некоторых пакетах (например, Norton AntiVirus и MacAfee VimsScan) используют метод, известный как сканирование по расписанию, для выполнения поиска вирусов на жестком диске в заданные периоды времени. Еще один метод заключается в использовании антивирусной программы в период простоя компьютера. Например, его можно использовать как часть программы экранной заставки.

Основные принципы компьютерной безопасности.

1. Обучите всех, кто пользуется вашим компьютером или сетью, основным принципам обеспечения компьютерной безопасности.
2. Установите антивирусную программу на компьютер. Установите на компьютер персональный брандмауэр.
3. Настройте почтовый клиент таким образом, чтобы он блокировал или помещал в отдельный каталог все потенциально опасные вложения.

4. Не пользуйтесь дисками, дискетами, флеш-картами, которыми Вы пользовались в заражённых ПК, не проверив их на наличие вирусов и не вылечив их.

5. Не поддавайтесь на сомнительные предложения в Интернете: просмотр интересного фильма или установка бесплатной программы и т.п.

6. Настройте свое антивирусное ПО таким образом, чтобы выполнялось регулярное обновление, как минимум раз в неделю.

7. Используйте авторитетные источники информации о компьютерных вирусах и «ложных тревогах».

8. Пользуйтесь программами для резервного копирования данных. Разработайте план восстановления системы на случай вирусной атаки.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Посмотрите, какие антивирусные программы установлены на Вашем ПК.

2. Откройте программу ESET NOD32 Antivirus и изучите окно программы (Рис.6).

3. Почитайте информацию на вкладках: Состояние защиты, Обновление, Настройка, Служебные программы, Справка и поддержка.

4. Посмотрите на вкладке Настройка, все ли опции включены: Защита в режиме реального времени, Защита электронной почты, Защита доступа в Интернет.

5. Включите вкладку Сканирование ПК. Выберите выборочное сканирование. Просканируйте диск локальный D.

6. Пока идёт сканирование, изучите содержимое вкладки Сервис. Какие файлы были помещены на карантин?

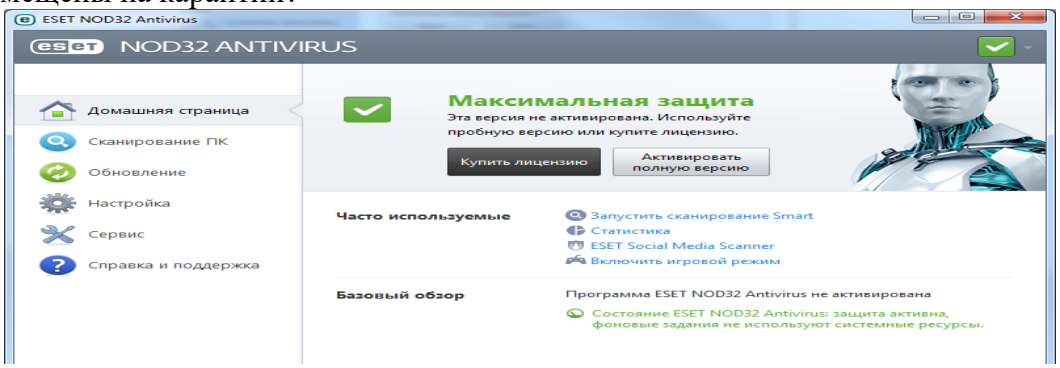


Рис.6

7. Результаты сканирования диска и дискеты запишите в отчёт.

8. В разделе Справочной системы программы найдите информацию о том, какие три уровня очистки поддерживает программа и запишите эту информацию в отчёт.

9. Изучите раздел справки Работа с ESET NOT32 Antivirus.

10. Изучите раздел справки Предупреждения и уведомления.

11. В служебных программах в Планировщике почитайте, какие задачи запланированы на ближайшее время и запишите эту информацию в отчёт.

Контрольные вопросы.

1. Что такое вирус?
2. Какие разновидности вирусов Вы знаете?
3. Как вирусы классифицируются по среде обитания?
4. Как вирусы классифицируются по степени вредного воздействия?
5. Какие виды вредоносных программ Вы знаете?
6. Как вирусы маскируются?
7. Когда обнаружили первый вирус?
8. Как Вы думаете, зачем изобретают вирусы?
9. Какие действия могут выполнять антивирусные программы?
10. Какие три задачи должна выполнять антивирусная программа?
11. Как обеспечить безопасность своей информации?

Лабораторная работа №9. Защита программ от несанкционированной эксплуатации и копирования

Цель работы: закрепить знания о защите программ от несанкционированного доступа к информации

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР9_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Под **несанкционированным доступом к информации** (НСД) 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;

- 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 1

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»

Примеры

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: $C=n^k=10^6$

Примем скорость перебора s=10 паролей в секунду. Получаем время перебора всех паролей t= C/s=10^5секунд=1667минут=28часов=1,2 дня.

Примем, что после каждого из m=3 неправильно введенных паролей идет пауза в v=5 секунд. Получаем время перебора всех паролей T=t*5/3=16667секунд=2778минут=46часов=1,9 дня.

$T_{итог}$ = t+T = 1,2 + 1,9 = 3,1 дня

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=log n C= lg C.

Определим количество вариантов C= t * s=10лет*10 паролей в сек. = 10*10*365*24*60*60=3,15* 10^9 вариантов

Таким образом, получаем длину пароля: k=lg (3,15*10^9) = 9,5

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

Практическая часть

Задание №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

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
11	33	15	30	0	0
12	28	7	20	3	0
13	77	11	20	0	0
14	55	7	15	5	10
15	145	9	200	7	15

Задание №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
11	45	50	30
12	64	40	20
13	74	100	15
14	66	100	45
15	63	95	200

Задание №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
11	6	90	100
12	11	50	200
13	10	80	30
14	7	120	15
15	9	80	50

Контрольные вопросы:

1. Что понимается под НСД?
2. Что такое парольная система.
3. Учетная запись пользователя – это...

Лабораторная работа № 10. Реализация системы защиты ПС

Цель работы: закрепить умение реализации электронно-цифровой подписи на примере RSA и изучить шифрования исходящей почты программным средством PGP

Порядок выполнения работы:

1. Изучить теоретический материал
2. Выбрать язык программирования
3. Выполнить практическую часть.
4. Оформить отчет. Отчет должен содержать:
   - наименование и цель работы;
   - ответы на контрольные вопросы;
   - код на выбранном языке программирования соответствующих заданий и скриншоты выполнения написанных программ;
   - последовательность выполненных действий задания 2;
   - выводы к выполненной лабораторной работе.

Отчет переименовать следующим образом: **ЗКИ_ЛР10_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Реализация элементов ЭЦП RSA

Основные понятия

Общие сведения

Протоколы ЭЦП с одной стороны относят к протоколам аутентификации, т.к. гарантируют, что сообщение поступило от достоверного отправителя, а с другой стороны к протоколам контроля целостности, т.к. гарантируют, что сообщение пришло в неискаженном виде. Более того, получатель в дальнейшем может использовать ЭЦП как доказательство достоверности сообщения третьим лицам (арбитру) в том случае, если отправитель впоследствии попытается отказаться от него.

Говоря о схеме цифровой подписи, обычно имеют в виду следующую классическую ситуацию:

- отправитель знает содержание сообщения, которое он подписывает;
- получатель, зная открытый ключ проверки подписи, может проверить правильность подписи полученного сообщения в любое время без какого-либо разрешения и участия отправителя;
- безопасность схемы подписи гарантируется.

Электронная цифровая подпись – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе (Федеральный закон "Об электронной цифровой подписи").

При создании цифровой подписи по классической схеме отправитель:

- применяет к исходному сообщению **T** хеш-функцию **h(T)** и получает хеш-образ r сообщения;
- вычисляет цифровую подпись **s по хеш-образу r с использованием своего закрытого ключа**;
- посылает сообщение **T** вместе с цифровой подписью s получателю.

Получатель, отделив цифровую подпись от сообщения, выполняет следующие действия:

- применяет к полученному сообщению **T** хеш-функцию **h(T)** и получает хеш-образ r сообщения;
- расшифровывает хеш-образ **r'** из цифровой подписи s с использованием открытого ключа отправителя;

- проверяет соответствие хеш-образов r и r’ и если они совпадают, то отправитель действительно является тем, за кого себя выдает, и сообщение при передаче не подверглось искажению.

Как видно из этой схемы, порядок использования ключей обратный тому, который используется при передаче секретных сообщений. Вначале отправитель использует свой закрытый ключ, а затем получатель применяет открытый ключ отправителя.

Разновидности ЭЦП

Кроме классической схемы ЭЦП различают еще несколько специальных:

- схема "конфиденциальной" (неотвергаемой) подписи – подпись не может быть проверена без участия сгенерировавшего ее лица;
- схема подписи "вслепую" ("затемненной" подписи) - отправитель не знает подписанного им сообщения;
- схема "мультиподписи" - вместо одного отправителя сообщение подписывает группа из нескольких участников;
- схема "групповой" подписи - получатель может проверить, что подписанное сообщение пришло от члена некоторой группы отправителей, но не знает, кем именно из членов группы оно подписано. В тоже время, в случае необходимости, отправитель может быть определен;
- и др.

Этап 1. Выработка ключей (выполняет отправитель **A**) - см. ЛР «Шифрование методом RSA».

Этап 2. Отправка сообщения и электронной подписи (выполняет отправитель **A**).

Отправка сообщения и ЭЦП на базе алгоритма RSA

Этап 3. Получение сообщения и проверка электронной подписи (выполняет получатель **B**).

№ п/п	Описание операции	Пример
1	Вычисление хеш-образа **h** = h(T), где T – исходное сообщение, h(T) – хеш-функция (для MD5 длина хеш-образа 128 бит).	h = 7
2	Выработка цифровой подписи **s** = $h^d \bmod n$, где d – закрытый ключ отправителя **A**, n – часть открытого ключа отправителя **A**.	$s = 7^{29} \bmod 91 = 63$
3	Отправка получателю **B** исходного сообщения **T** и цифровой подписи **s**.	

Получение сообщения и проверка ЭЦП на базе алгоритма RSA

Шифрование передаваемых данных программным средством PGP

Основные понятия

Что такое PGP?

Прикладная криптосистема PGP (Pretty Good Privacy, Довольно хорошая секретность) была разработана и опубликована в интернете в 1991 году программистом и математиком Массачусетского Политеха Филиппом Циммерманом, по сути, оказавшись первым продуктом подобного уровня, представленным для свободного доступа всему миру (за что впоследствии Циммерман поплатился несколькими годами уголовного преследования со стороны Таможенной службы США — в то время экспорт стойких криптотехнологий за пределы Штатов был запрещён). Изначальной целью разработки была защита гражданских прав пользователей глобальной сети, а главной задачей программы стала криптографическая защита электронной почты — шифрование.

Программа основана на так называемой асимметричной криптографии, использующей взаимосвязные пары ключей: закрытый, хранящийся только у владельца для цели расшифрования данных и их цифрового подписания, и открытый, который не нуждается в защите, может быть широко распространен и используется для зашифрования и сличения цифровых подписей (все эти уникальные возможности достигаются засчёт особого математического аппарата). Это идеальное решение для людей, не имеющих существующего согласованного тайного

ключа. Вы можете взять открытый ключ адресата из любого открытого источника, с его Интернет-сайта, например, зашифровать сообщение и отправить. Никто, кроме получателя с соответствующим закрытым ключом, не сумеет прочитать ваше письмо.

С тех пор PGP претерпел значительные изменения и преобразился, как согласно духу времени и новых угроз, так и вследствие того, что теперь значительную часть пользователей программы составляют не только обычные люди, но и крупные организации и бизнесы. Сегодня PGP — этонесколько линеек приложений, различающихся назначением и перечнем решаемых задач, функциональностью, принципами работы и средой исполнения, но объединённых полной совместимостью благодаря стандарту OpenPGP, а также своей исключительной надёжностью в обеспечении защиты информации.

№ п/п	Описание операции	Пример
1	Вычисление хеш-образа по полученному сообщению **h'** = h(T'), где T' – полученное сообщение. Если T = T', то должно быть h = h'.	h' = 7
2	Вычисление хеш-образа из цифровой подписи **h"** = $s^e \bmod n$, где e и n – открытый ключ отправителя **A**.	h" = $63^5 \bmod 91 = 7$
3	Т.к. h' = h", то получатель **B** делает вывод, что полученное сообщение T' = T и оно действительно отправлено **A**.	

Практическая часть

Задание №1

В лабораторной работе необходимо привести последовательность выполнения процедур генерации и проверки ЭЦП.

Посчитать контрольную сумму своего ФИО методом md5. На базе алгоритма RSA получить ЭЦП. Удостоверится что ЭЦП принадлежит именно этому сообщению.

Задание №2

1. Установите почтовый клиент, например, Mozilla Thunderbird.
2. Настройте этот почтовый клиент для своего почтового ящика.
3. Установите дополнение которое позволяет шифровать данные Enigmail.
4. При установке дополнения предложит установить ядро шифрования Gpg4win.
5. Сгенерируйте пару ключей.
6. Отправьте открытый ключ на почту ****,ФИО и номер группы укажите в теме письма.

Контрольные вопросы:

1. Дайте определение понятию «электронная цифровая подпись».
2. Опишите последовательность действий участников протокола при отправке и проверке ЭЦП.
3. Какой порядок использования ключей (открытый; закрытый) при отправке и проверке ЭЦП?
4. Опишите схему протокола ЭЦП на основе алгоритма RSA.
5. Перечислите специальные схемы ЭЦП

Лабораторная работа № 11. Защита данных средствами СУБД MS Access

Цель работы: закрепить знания о создании зашифрованных дисков программным средством TrueCrypt

Порядок выполнения работы:

Изучить теоретический материал. В лабораторной работе необходимо создать виртуальный зашифрованный диск. Наполнить его небольшим количеством файлов. Создать скрытый том. Попробовать прочесть созданный файл тома. Размонтировать и снова смонтировать том.

В отчет вставить скриншоты выполненных действий.

Отчет переименовать следующим образом: **ЗКИ_ЛР11_Фамилия_№группы** и сдать на проверку преподавателю.

Теоретическая часть

Создание виртуальных зашифрованных дисков (программное средство TrueCrypt)

Программа TrueCrypt позволяет создавать полностью зашифрованный виртуальный диск, на котором после начального ввода пароля доступа данные будут автоматически зашифровываться и расшифровываться без вашего вмешательства.

TrueCrypt является международным проектом - некоммерческой свободно распространяемой программой с открытым исходным кодом, проверенным независимыми экспертами.

Защищенный виртуальный диск создается в виде файла, который можно скопировать на любой компьютер и смонтировать, введя пароль. Формат файла вы в процессе создания зашифрованного диска можете выбрать сами в зависимости от необходимого вам размера диска.

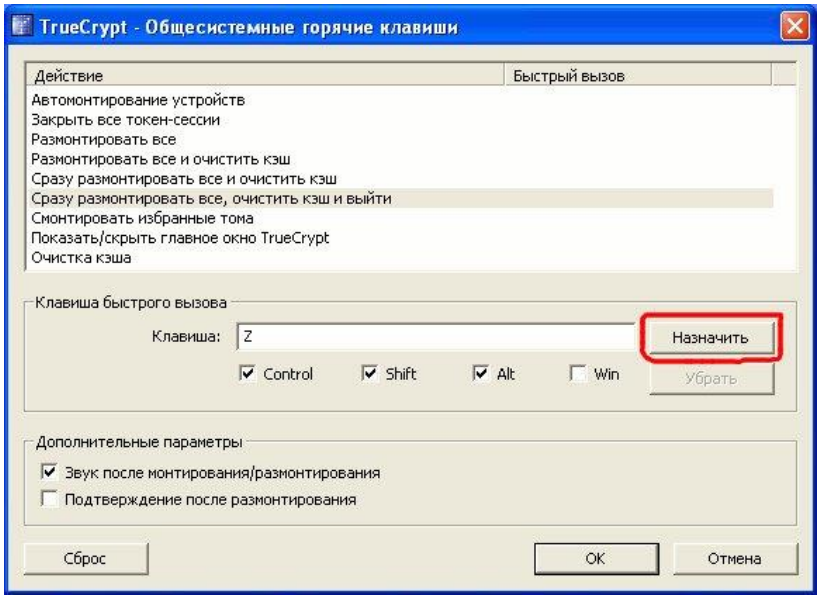
Еще одно достоинство данной программы в том, что ни один диск TrueCrypt не может быть идентифицирован. Защищенный диск, созданный с помощью TrueCrypt, невозможно отличить от набора случайных данных, то есть файл нельзя связать с TrueCrypt как с программой, его создавшей.

Дополнительно внутри данного диска (в терминологии TrueCrypt - внешнего тома) можно создать так называемый скрытый диск размером меньше оригинала (скрытый том). Идея заключается в том, что во внешнем томе вы можете разместить абсолютно нейтральные файлы, которые не содержат конфиденциальной информации. А вот на скрытом томе вы будете размещать именно конфиденциальную информацию. Решение, какой том монтировать, каждый раз принимаете вы сами, и оно зависит от применения того или иного пароля (у внешнего и скрытого томов свои пароли). В случае монтирования внешнего тома скрытый том с конфиденциальными файлами остается невидим.

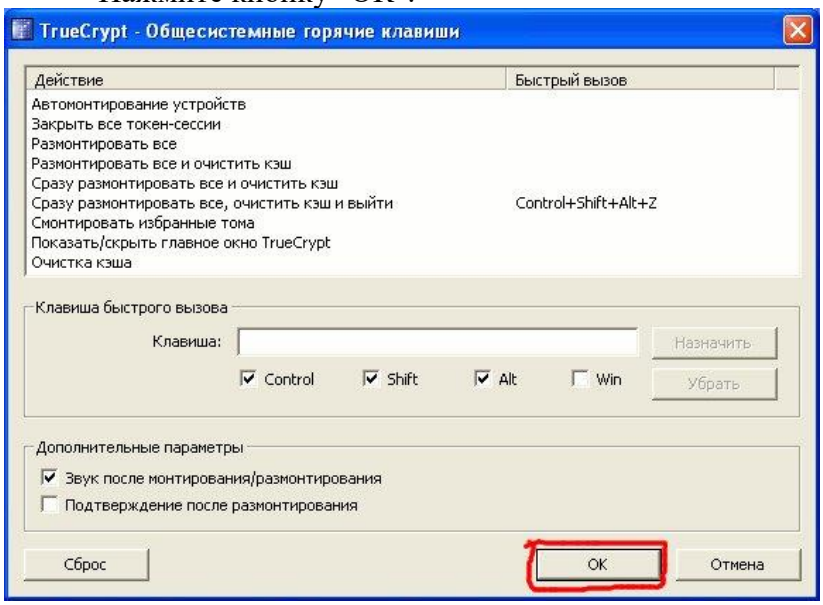
Настройка TrueCrypt

Для начала сконфигурируем TrueCrypt таким образом, чтобы можно было закрыть программу, размонтировать диски и очистить временный буфер моментально по сигналу тревоги. Для этого выполните следующие действия:

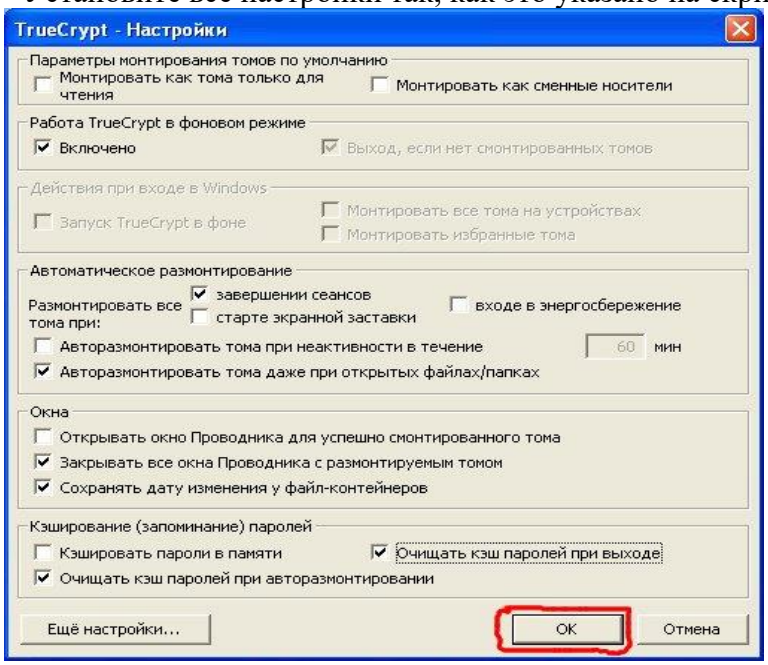
- Пройдите в меню программы Настройки - Горячие клавиши... - В открывшемся окне выберите пункт "Сразу размонтировать все, очистить кэш и выйти".
- Установите курсор в поле "Клавиша" и нажмите на клавиатуре клавишу, которую вам удобно рассматривать как "горячую". В нашем примере это клавиша с буквой "Z". Вы также можете выбрать, какие клавиши использовать совместно с ней - Control, Shift, Alt, Win, установив или убрав флажок рядом с названием клавиши. Мы выбрали клавиши Control, Shift и Alt. Это означает, что при нажатии одновременно на клавиши Control, Shift, Alt и Z все TrueCrypt-диски будут размонтированы, временный буфер очищен и программа закроется.
- После выбора "горячих" клавиш нажмите кнопку "Назначить".



- Нажмите кнопку "ОК".



Затем установите остальные настройки:
- Пройдите в меню программы Настройки - Настройки...
- Установите все настройки так, как это указано на скриншоте. Нажмите кнопку "ОК".

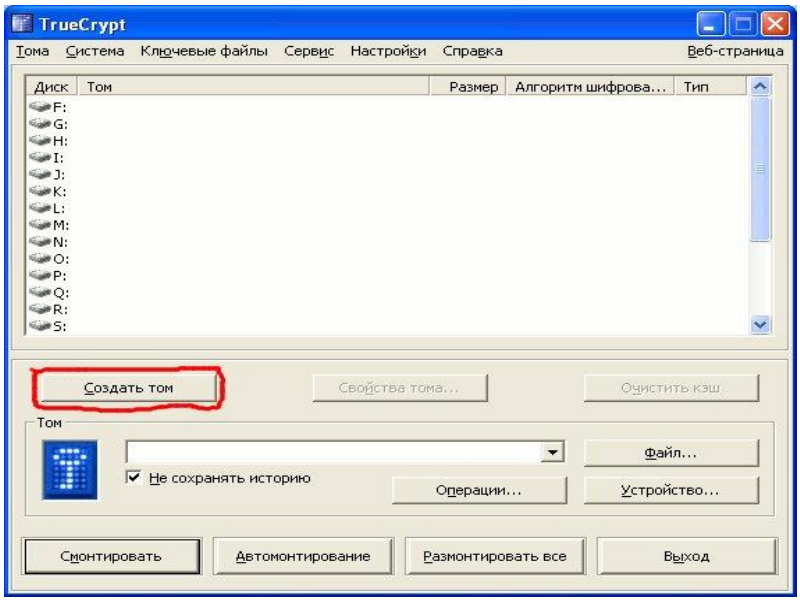


Работа с программой TrueCrypt

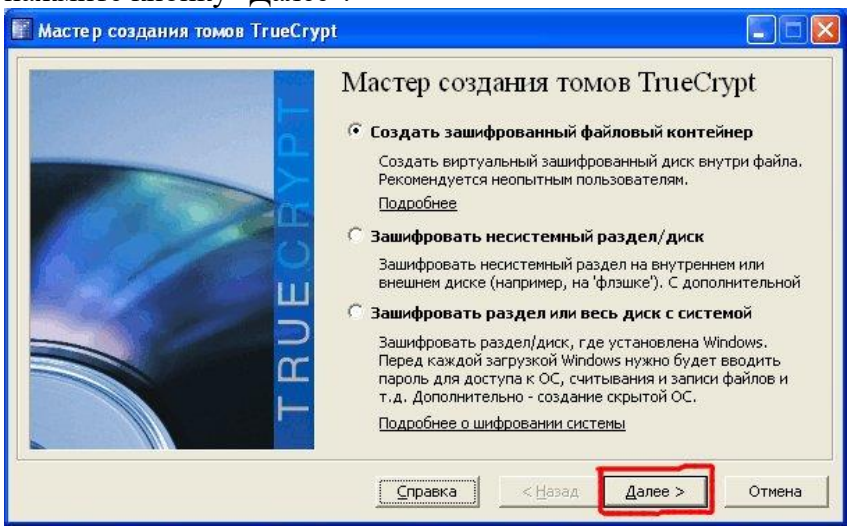
Итак, перейдем непосредственно к работе с программой. Для начала научимся создавать внешний том. Затем - скрытый том. После этого научимся монтировать и размонтировать тома.

Создание внешнего тома TrueCrypt

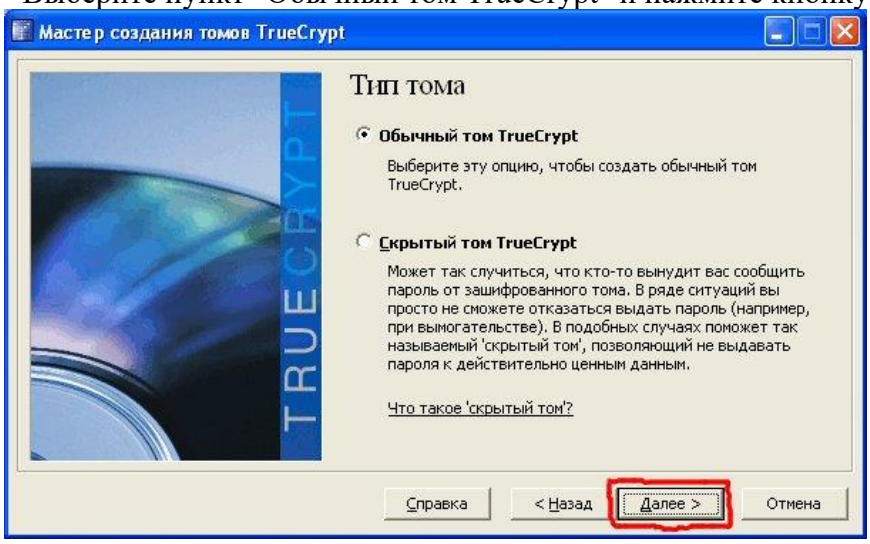
- В основном окне программы нажмите кнопку "Создать том". Запустится мастер создания томов.



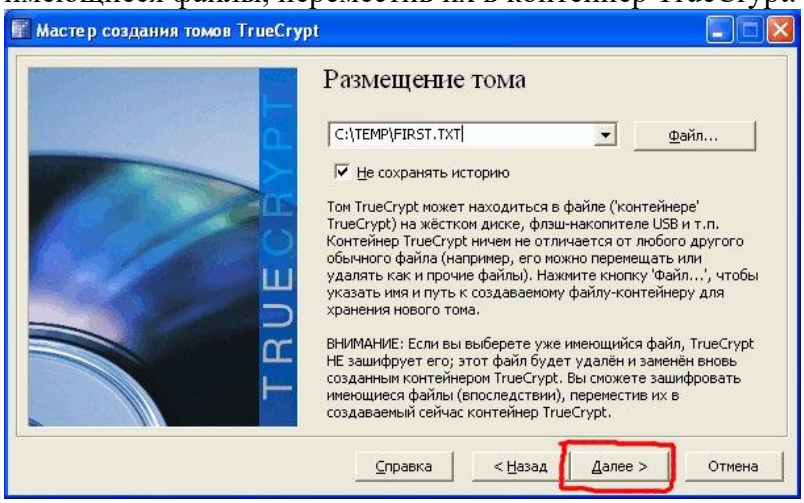
- В появившемся окне выберите пункт "Создать зашифрованный файловый контейнер" и нажмите кнопку "Далее".



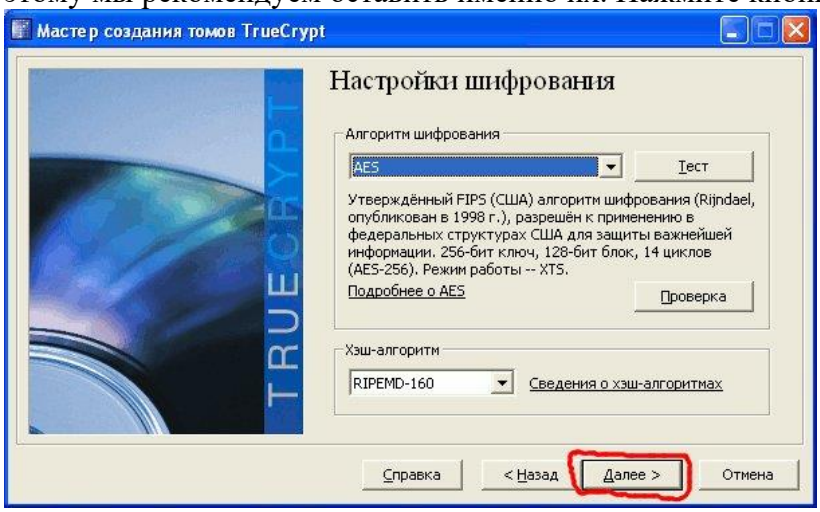
- Выберите пункт "Обычный том TrueCrypt" и нажмите кнопку "Далее".



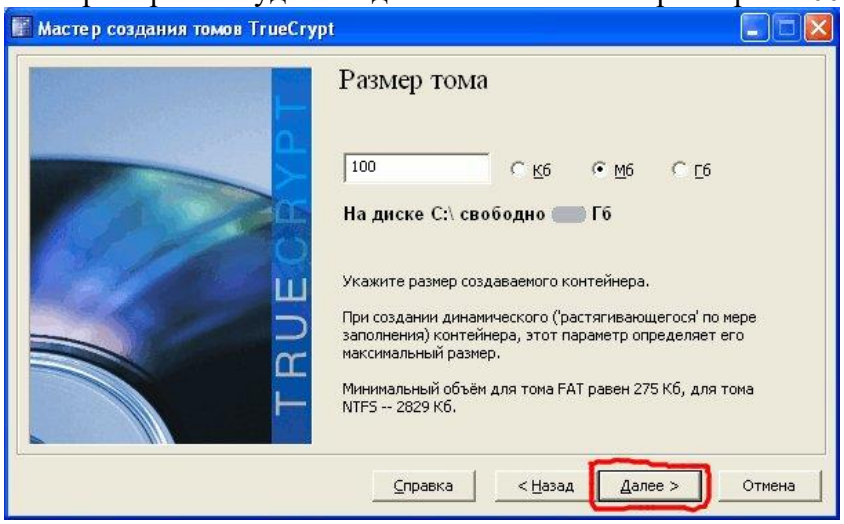
- Введите путь размещения файла, его имя и расширение. В нашем примере это C:\TEMP\FIRST.TXT. Этот файл будет создан и именно под него будет замаскирован том TrueCrypt. Убедитесь, что установлен флажок "Не сохранять историю". Нажмите кнопку "Далее". Если вы выберете уже имеющийся файл, то TrueCrypt не зашифрует его; этот файл будет удален и заменен созданным контейнером TrueCrypt. Впоследствии вы сможете зашифровать имеющиеся файлы, переместив их в контейнер TrueCrypt.



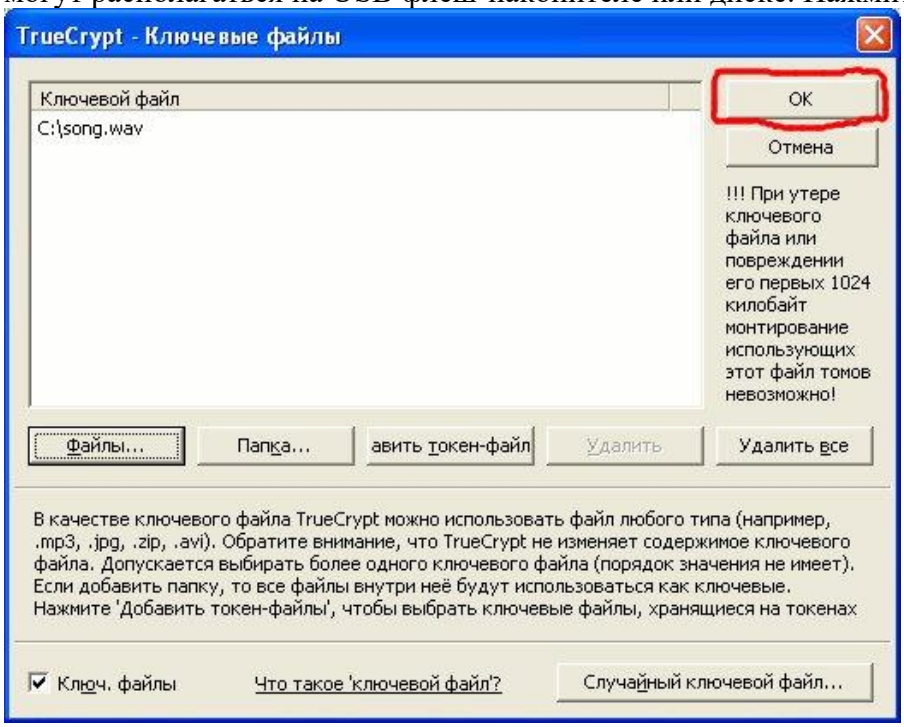
- В следующем окне вы можете выбрать алгоритм шифрования (по умолчанию - AES) и хэш-алгоритм (по умолчанию - RIPEMD-160). Установленные по умолчанию значения вполне надежны и обеспечивают приемлемую скорость шифрования/дешифрования информации. Поэтому мы рекомендуем оставить именно их. Нажмите кнопку "Далее".



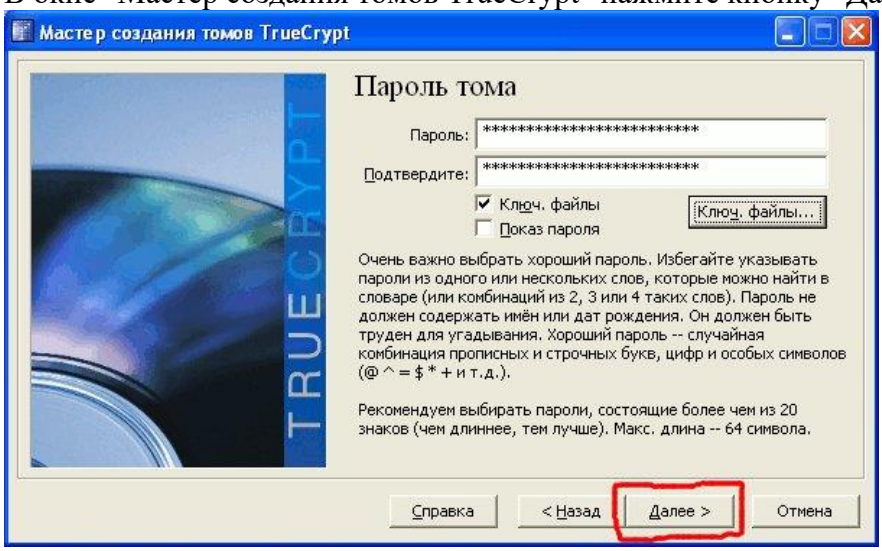
- Выберите размер тома и нажмите кнопку "Далее". При этом учтите, что скрытый том TrueCrypt, который мы будем создавать впоследствии, будет меньше, чем внешний том. В данном примере мы будем создавать внешний том размером 100 Мб.



- Введите пароль и подтвердите его. Максимальная длина пароля - 64 символа. Постарайтесь выбрать пароль, состоящий не менее, чем из 20 знаков.
Также мы рекомендуем выбрать ключевые файлы, т.е. файлы, которые вам нужно будет указывать каждый раз, когда вы будете монтировать том TrueCrypt. В качестве ключевых файлов подойдут любые, их расширение значения не имеет. Однако впоследствии вам нужно будет следить за тем, чтобы файлы, выбранные вами в качестве ключевых, не были повреждены. Иначе вы не сможете смонтировать том TrueCrypt и получить доступ к данным.
Для выбора ключевых файлов установите флажок "Ключ. файлы" и нажмите кнопку "Ключ. файлы...". В открывшемся окне нажмите кнопку "Файлы..." и выберите ключевые файлы. При этом учтите, что запоминается путь, а не только имена файлов. Для того, чтобы том мог быть смонтирован, файлы должны располагаться в той же директории, что и при их выборе в качестве ключевых во время создания тома. Если добавить папку, нажав на кнопку "Папка", то все файлы внутри нее будут использоваться как ключевые.
В нашем примере мы выбрали файл song.wav. Полный путь к файлу, который будет указан - C:\song.wav. Вы можете выбрать любое количество файлов. Для большей секретности файлы могут располагаться на USB флеш-накопителе или диске. Нажмите кнопку "ОК".

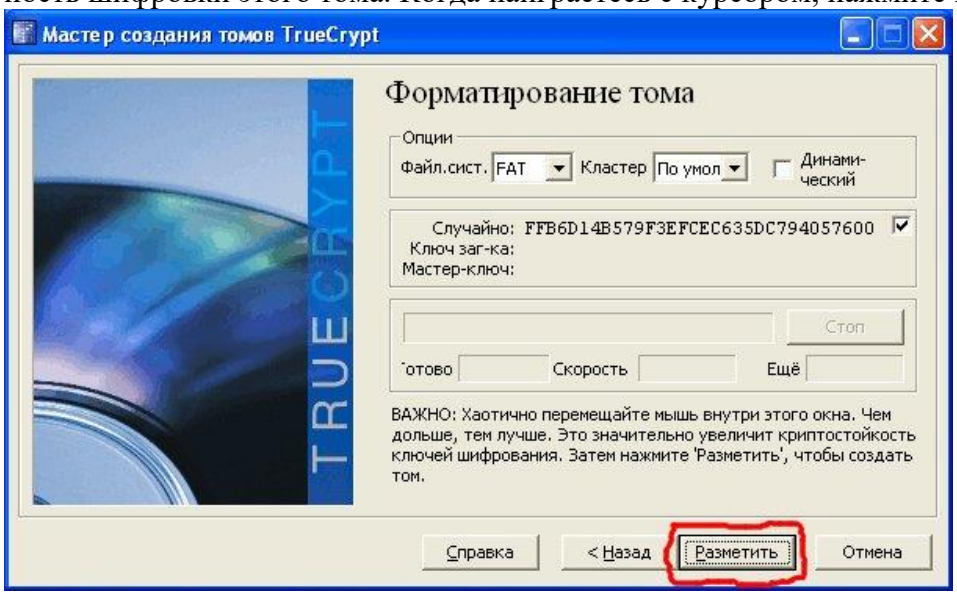


В окне "Мастер создания томов TrueCrypt" нажмите кнопку "Далее".

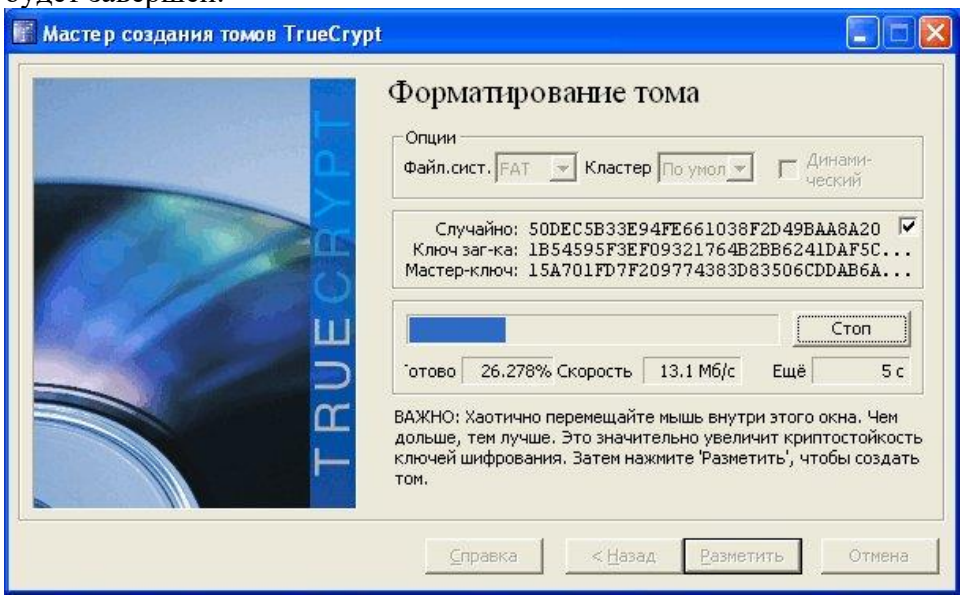


- В следующем окне вы можете выбрать тип файловой системы создаваемого тома. Для внешнего тома мы рекомендуем использовать файловую систему FAT, чтобы скрытый том можно было создать максимально возможного размера. Тип кластера установите "По умолчанию". Не устанавливайте флажок "Динамический".

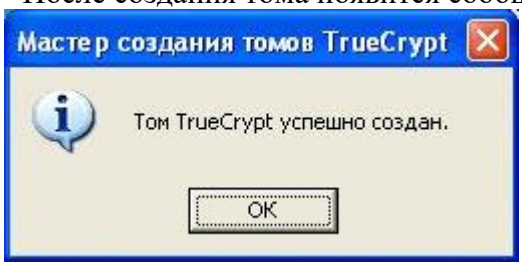
После этого поводите курсором мыши по этому окну. Вы заметите, что при движении курсора случайный код меняется быстрее. Такое броуновское движение курсором улучшает случайность шифровки этого тома. Когда наиграетесь с курсором, нажмите кнопку "Разметить".



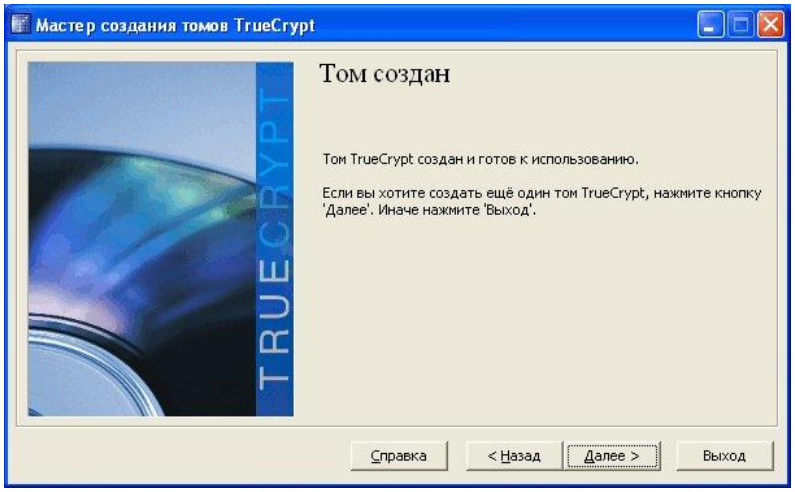
- После этого будет запущен процесс создания внешнего тома TrueCrypt. Дождитесь, пока он будет завершен.



- После создания тома появится сообщение о его успешном создании. Нажмите кнопку "ОК".



- В следующем окне нажмите кнопку "Далее", если хотите продолжить процесс создания томов, или кнопку "Выход".

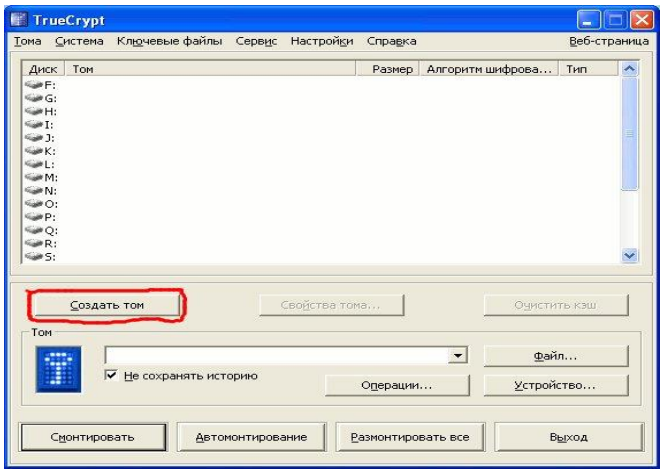


4.2. Создание скрытого тома TrueCrypt

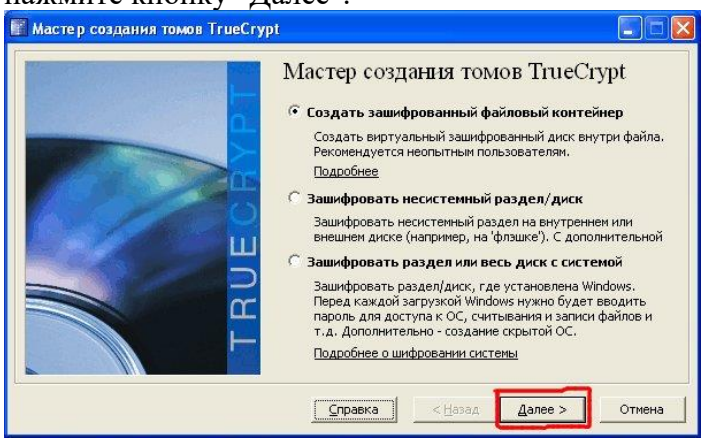
После того, как создан внешний том TrueCrypt, можно перейти к созданию скрытого тома. Обратите внимание, что скрытый том будет спрятан внутрь только что созданного внешнего тома. Отсюда следует, что скрытый том не может быть больше внешнего тома. В нашем примере это 100 Мб для внешнего тома и 70 Мб для скрытого. Такая разница нужна для того, чтобы мы могли впоследствии разместить во внешнем томе несколько файлов с посторонней информацией для маскировки. Вы можете создать несколько скрытых томов внутри одного внешнего тома. Главное, чтобы суммарный объем всех скрытых томов не превышал размера внешнего тома.

Также обратите внимание, что для создания скрытого тома нужно ввести пароль, который вы ранее использовали для внешнего тома, а в окне "Пароль скрытого тома" - новый пароль для скрытого тома (дважды - в полях "Пароль" и "Подтвердите").

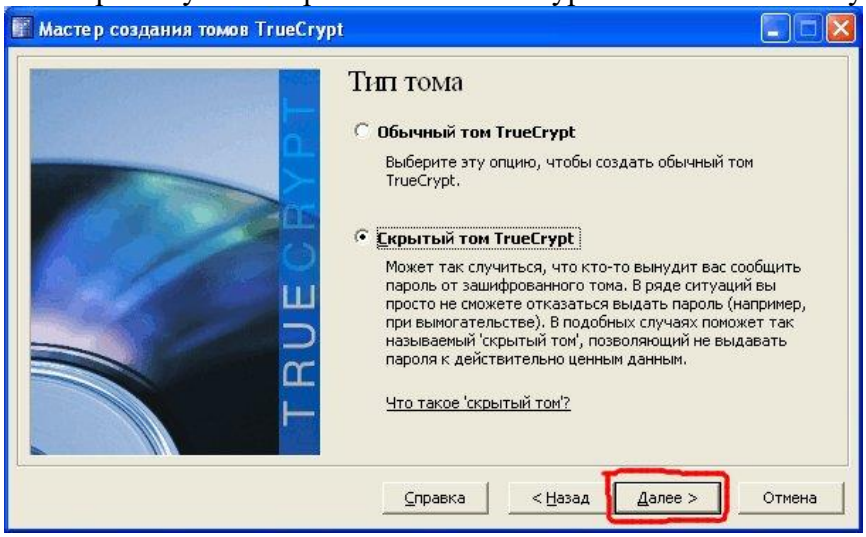
- В основном окне программы нажмите кнопку "Создать том". Запустится мастер создания томов.



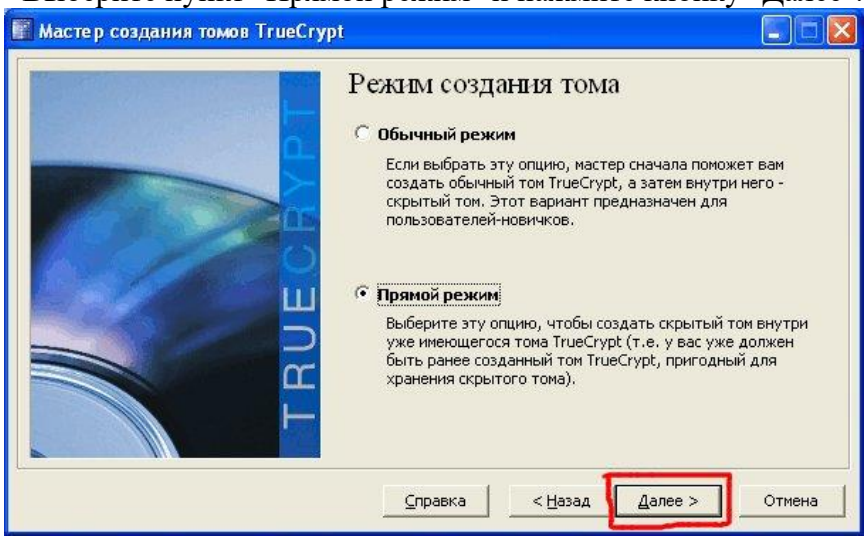
- В появившемся окне выберите пункт "Создать зашифрованный файловый контейнер" и нажмите кнопку "Далее".



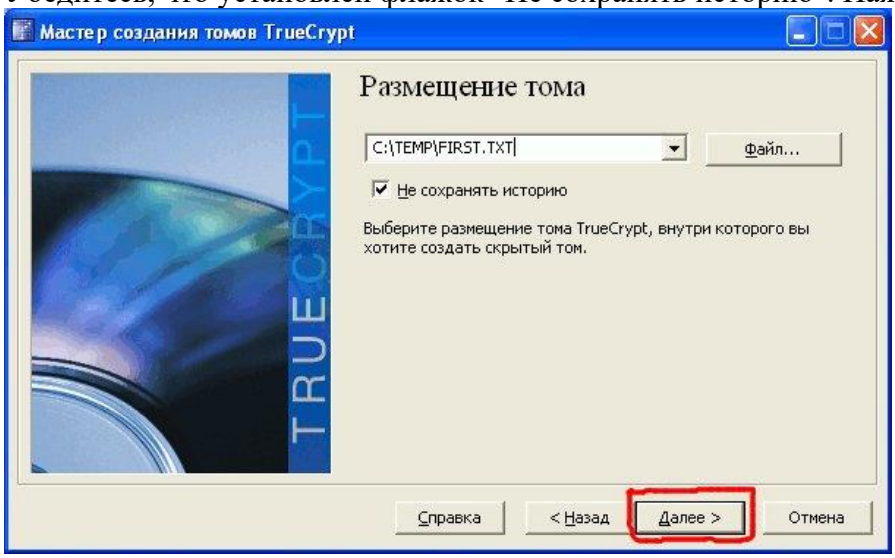
- Выберите пункт "Скрытый том TrueCrypt" и нажмите кнопку "Далее".



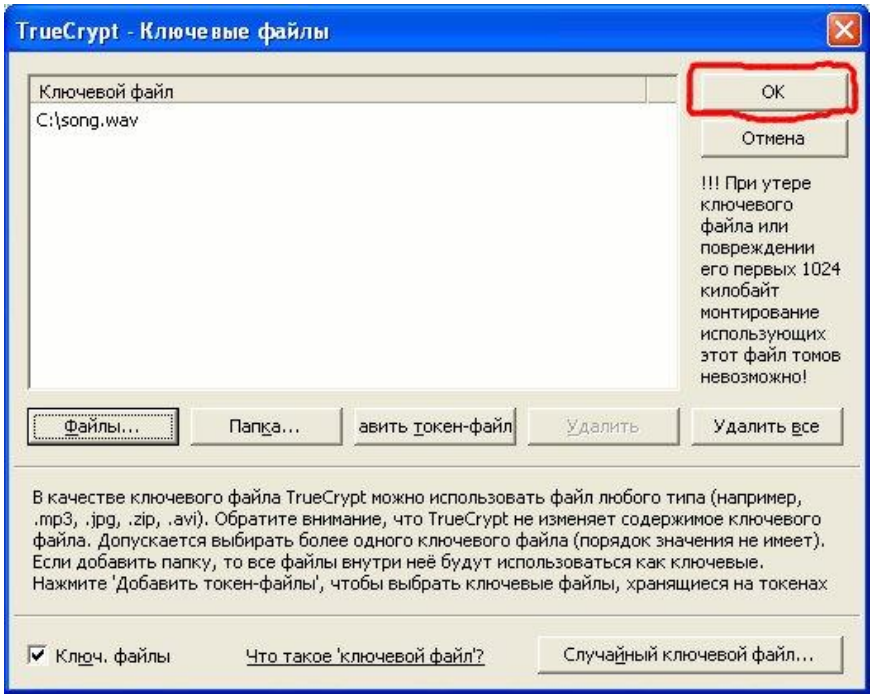
- Выберите пункт "Прямой режим" и нажмите кнопку "Далее".



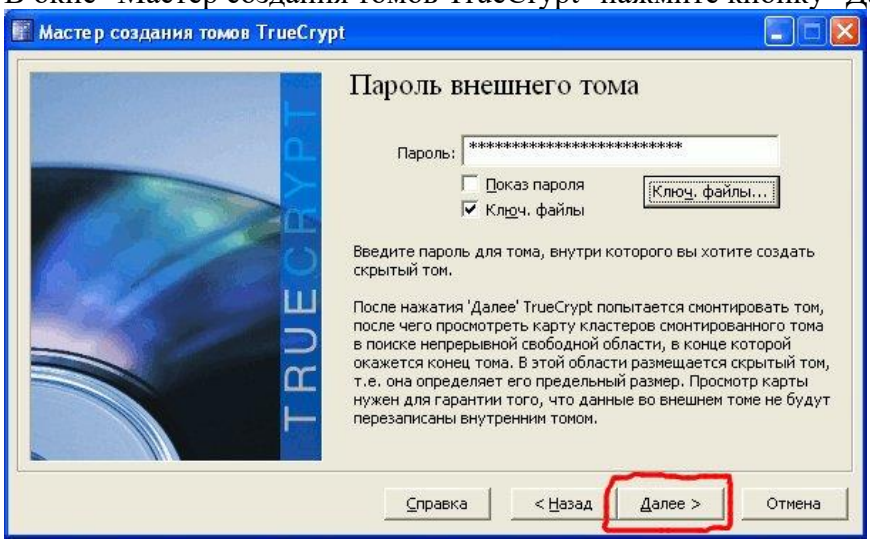
- Укажите путь к файлу, который маскирует внешний том TrueCrypt. Внутри этого внешнего тома будет создан скрытый том TrueCrypt. В нашем примере этот путь - C:\TEMP\FIRST.TXT. Убедитесь, что установлен флажок "Не сохранять историю". Нажмите кнопку "Далее".



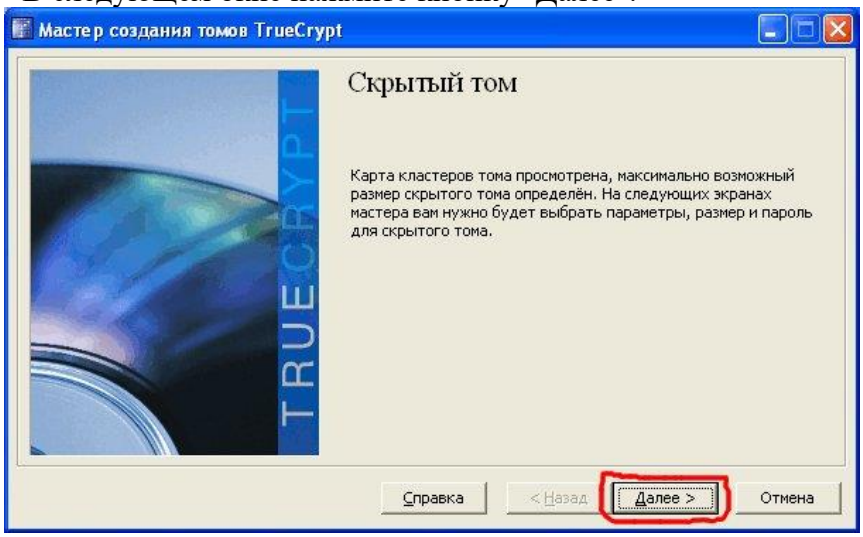
- В следующем окне введите пароль внешнего тома. Если вы при создании внешнего тома указывали ключевые файлы, необходимые для монтирования тома, то установите флажок "Ключ. файлы" и нажмите на кнопку "Ключ. файлы...". В открывшемся окне нажмите кнопку "Файлы..." и выберите файлы, которые вы указывали в качестве ключевых при создании внешнего тома. В нашем примере это C:\song.wav. После того, как будут указаны все ключевые файлы, нажмите кнопку "ОК".



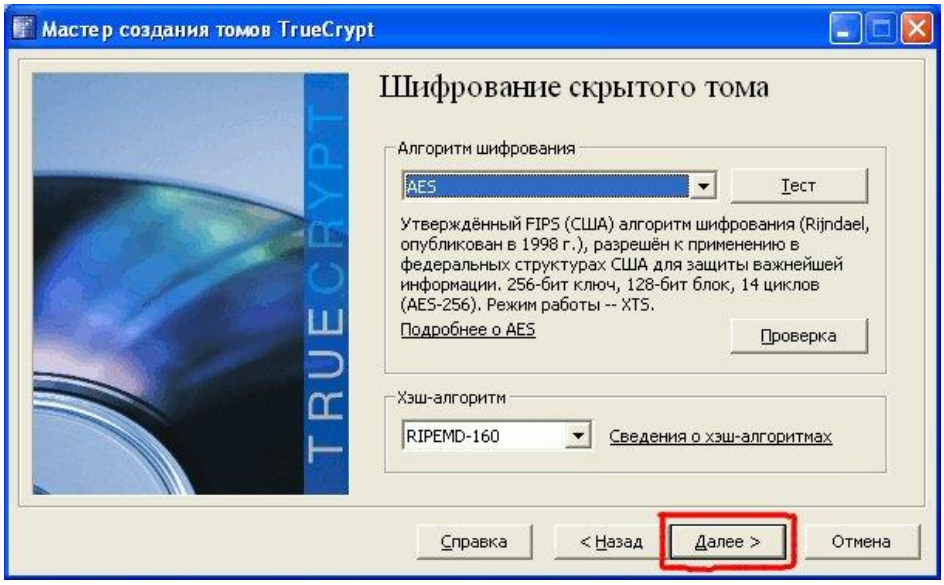
В окне "Мастер создания томов TrueCrypt" нажмите кнопку "Далее".



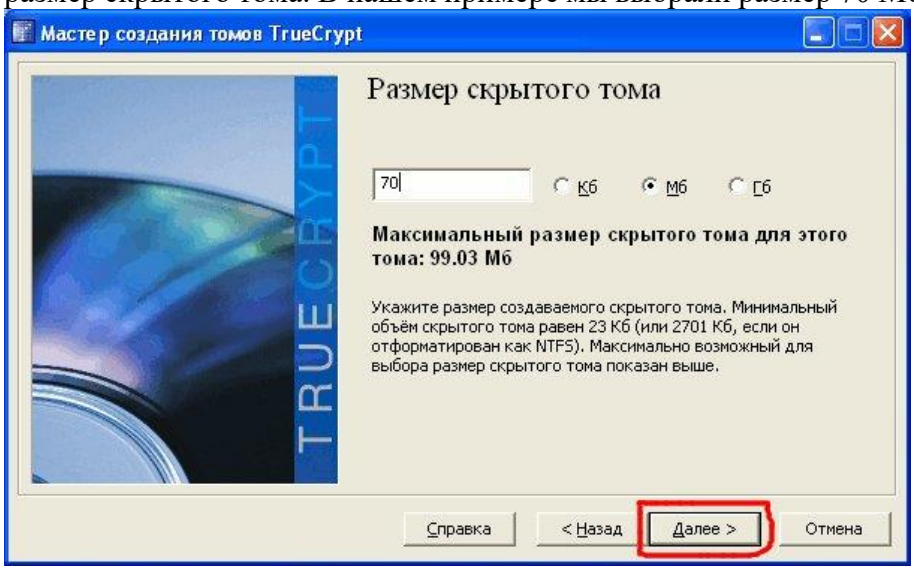
- В следующем окне нажмите кнопку "Далее".



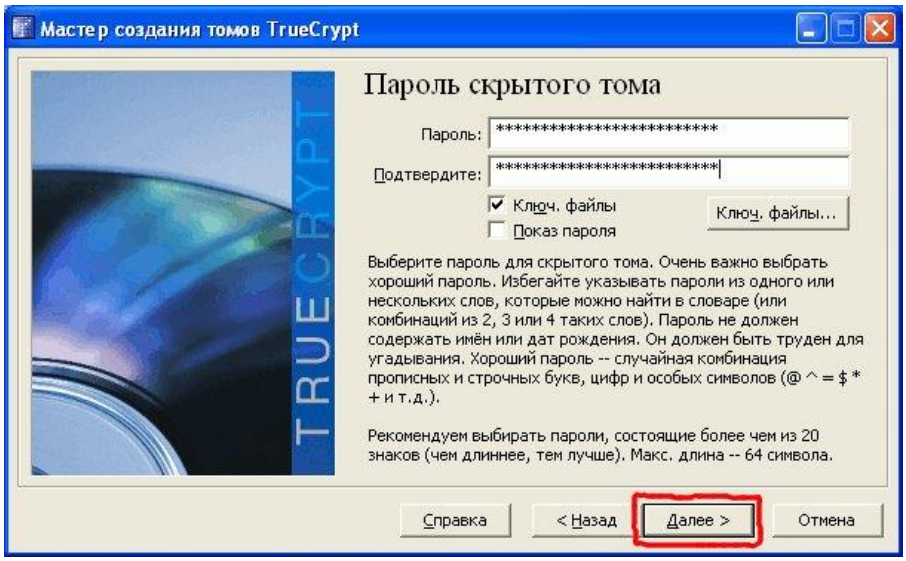
- В следующем окне вы можете выбрать алгоритм шифрования и хэш-алгоритм скрытого тома. Мы рекомендуем оставить те значения, которые установлены по умолчанию. Нажмите кнопку "Далее".



- В следующем окне будет указан максимально возможный размер скрытого тома. Укажите размер скрытого тома. В нашем примере мы выбрали размер 70 Мб. Нажмите кнопку "Далее".

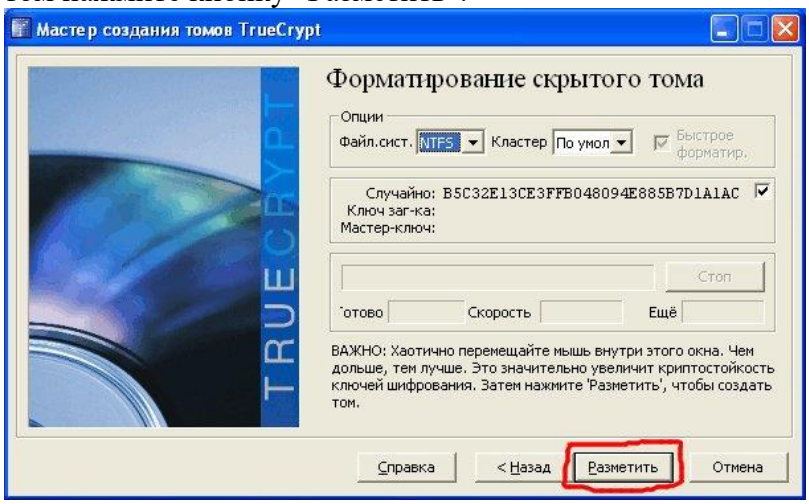


- Введите пароль скрытого тома и подтвердите его. Максимальная длина пароля - 64 символа. Постарайтесь выбрать пароль, состоящий не менее, чем из 20 знаков. Также выберите и укажите ключевые файлы так, как вы это уже умеете. Нажмите кнопку "Далее".

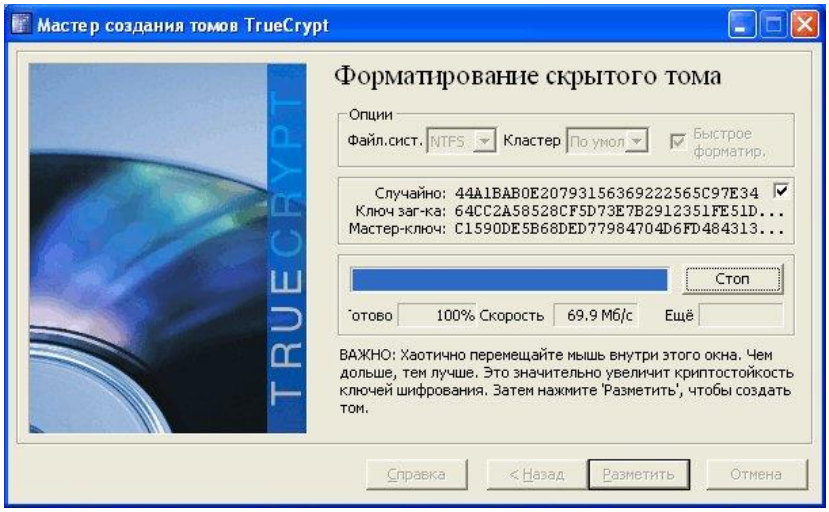


- В следующем окне вы можете выбрать тип файловой системы создаваемого тома. Для скрытого тома мы рекомендуем использовать файловую систему NTFS. Тип кластера установите "По умолчанию".

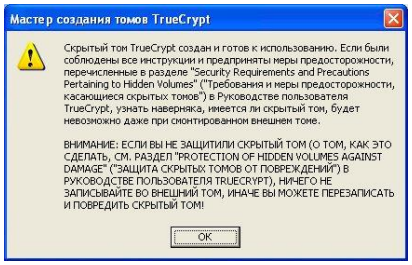
После этого поводите курсором мыши по этому окну, как и при создании внешнего тома. Потом нажмите кнопку "Разметить".



Будет запущен процесс создания скрытого тома TrueCrypt. Дождитесь, пока он будет завершен.



- После окончания процесса создания скрытого тома появится окно, предупреждающее о разумности защиты скрытых томов при монтировании внешнего тома. Если вы об этом забудете, вы можете случайно затереть информацию на скрытом томе, когда размещаете файлы на внешнем томе. В этом окне нажмите кнопку "ОК".



- В следующем окне нажмите кнопку "Выход".