Лабораторные и практические работы по курсу «Информационная безопасность»

Перечень лабораторных работ:

1. Парольная защита.
2. Архивирование с паролем.
3. Шифр простой замены, таблица Вижинера.
4. Обмен ключами по Диффи-Хелману.
5. Шифр RSA.
6. Циклические коды.

В отчете по лабораторной работе должно содержаться:

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

Рейтинг

№	час	баллы	Наименование
1	2	60	Парольная защита
2	2	80	Архивирование с паролем
3	4	200	Шифр простой замены, таблица Вижинера
4	2	100	Обмен ключами по Диффи-Хелману
5	4	230	Шифр RSA
6	2	80	Циклические коды
7	2		Итоговое
	18	750	

Лабораторная работа 1

Парольная защита

Под **несанкционированным доступом к информации** (НСД) согласно руководящим документам Гостехкомиссии будем понимать доступ к информации, нарушающий установленные правила разграничения доступа и осуществляемый с использованием штатных средств, предоставляемых СВТ или АС. НСД может носить случайный или намеренный характер.

Можно выделить несколько обобщенных категорий методов защиты от НСД, в частности:

- организационные;
- технологические;
- правовые.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Учетная запись пользователя — совокупность его идентификатора и его пароля.

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

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

Основными компонентами парольной системы являются:

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

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

1. Разглашение параметров учетной записи через:
    - подбор в интерактивном режиме;
    - подсматривание;
    - преднамеренную передачу пароля его владельцем другому лицу;
    - захват базы данных парольной системы (если пароли не хранятся в базе в открытом виде, для их восстановления может потребоваться подбор или дешифрование);
    - перехват переданной по сети информации о пароле;
    - хранение пароля в доступном месте.
2. Вмешательство в функционирование компонентов парольной системы через:
    - внедрение программных закладок;
    - обнаружение и использование ошибок, допущенных на стадии разработки;
    - выведение из строя парольной системы.

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

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

- 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»

Принудительная смена пароля при первой регистрации пользователя в системе	Защищает от неправомерных действия системного администратора, имеющего доступ к паролю в момент создания учетной записи

2. Примеры.

Например 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. Пример 2.
Определить минимальную длину пароля, алфавит которого состоит из 10 символов, время перебора которого было не меньше 10 лет.
Алфавит составляют символы n=10.
Длина пароля рассчитывается: $k=\log_n C= \lg C$.
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.

3. Задания.

1. Определить время перебора всех паролей с параметрами.
Алфавит состоит из n символов.
Длина пароля символов k.
Скорость перебора s паролей в секунду.
После каждого из m неправильно введенных паролей идет пауза в v секунд

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

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

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

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

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

Лабораторная работа 2

Архивирование с паролем

1. Теория.

Архиваторы – это программы для создания архивов. Архивы предназначены для хранения данных в удобном компактном виде. В качестве данных обычно выступают файлы и папки. Как правило, данные предварительно подвергаются процедуре сжатия или упаковки. Поэтому почти каждый архиватор одновременно является программой для сжатия данных. С другой стороны, любая программа для сжатия данных может рассматриваться как архиватор. Эффективность сжатия является важнейшей характеристикой архиваторов. От нее зависит размер создаваемых архивов. Чем меньше архив, тем меньше места требуется для его хранения. Для передачи нужна меньшая пропускная способность канала передачи или затрачивается меньшее время. Преимущества архивов очевидны, если учесть, что данные уменьшаются в размере и в 2 раза, и в 5 раз.

Сжатие данных используется очень широко. Можно сказать, почти везде. Например, документы PDF , как правило, содержат сжатую информацию. Довольно много исполняемых файлов EXE сжаты специальными упаковщиками. Всевозможные мультимедийные файлы (GIF , JPG , MP 3, MPG) являются своеобразными архивами.

Основным недостатком архивов является невозможность прямого доступа к данным. Их сначала необходимо извлечь из архива или распаковать. Операция распаковки, впрочем, как и упаковки, требует некоторых системных ресурсов. Это не мгновенная операция. Поэтому архивы в основном применяют со сравнительно редко используемыми данными. Например, для хранения резервных копий или установочных файлов.

В данный момент существует много архиваторов. Они имеют разную распространенность и эффективность. Некоторые интересные архиваторы не известны широкому кругу потенциальных пользователей. Особый интерес представляют оценка и сравнение эффективности сжатия популярных архиваторов.

Методы сжатия архиваторов.

Разработано большое количество разнообразных методов, их модификаций и подвидов для сжатия данных. Современные архиваторы, как правило, одновременно используют несколько методов одновременно. Можно выделить некоторые основные.

Кодирование длин серий (RLE - сокращение от run - length encoding - кодирование длин серий).

Очень простой метод. Последовательная серия одинаковых элементов данных заменяется на два символа: элемент и число его повторений. Широко используется как дополнительный, так и промежуточный метод. В качестве самостоятельного метода применяется, например, в графическом формате BMP .

Словарный метод (LZ - сокращение от Lempel Ziv - имена авторов).

Наиболее распространенный метод. Используется словарь, состоящий из последовательностей данных или слов. При сжатии эти слова заменяются на их коды из словаря. В наиболее распространенном варианте реализации в качестве словаря выступает сам исходный блок данных.

Основным параметром словарного метода является размер словаря. Чем больше словарь, тем больше эффективность. Однако для неоднородных данных чрезмерно большой размер может быть вреден, так как при резком изменении типа данных словарь будет заполнен неактуальными словами. Для эффективной работы данного метода при сжатии требуется дополнительная память. Приблизительно на порядок больше, чем нужно

для исходных данных словаря. Существенным преимуществом словарного метода является простая и быстрая процедура распаковки. Дополнительная память при этом не требуется. Такая особенность особенно важна, если необходим оперативный доступ к данным.

Энтропийный метод (Huffman - кодирование Хаффмена, Arithmetic coding - арифметическое кодирование)

В этом методе элементы данных, которые встречаются чаще, кодируются при сжатии более коротким кодом, а более редкие элементы данных кодируются более длинным кодом. За счет того, что коротких кодов значительно больше, общий размер получается меньше исходного.

Широко используется как дополнительный метод. В качестве самостоятельного метода применяется, например, в графическом формате JPG .

Метод контекстного моделирования (CM - сокращение от context modeling - контекстное моделирование)

В этом методе строится модель исходных данных. При сжатии очередного элемента данных эта модель выдает свое предсказание или вероятность. Согласно этой вероятности, элемент данных кодируется энтропийным методом. Чем точнее модель будет соответствовать исходным данным, тем точнее она будет выдавать предсказания, и тем короче будут кодироваться элементы данных.

Для построения эффективной модели требуется много памяти. При распаковке приходится строить точно такую же модель. Поэтому скорость и требования к объему оперативной памяти для упаковки и распаковки почти одинаковы. В данный момент методы контекстного моделирования позволяют получить наилучшую степень сжатия, но отличаются чрезвычайно низкой скоростью.

PPM (PPM - Prediction by Partial Matching - предсказание по частичному совпадению).

Это особый подвид контекстного моделирования. Предсказание выполняется на основании определенного количества предыдущих элементов данных. Основным параметром является порядок модели, который задает это количество элементов. Чем больше порядок модели, тем выше степень сжатия, но требуется больше оперативной памяти для хранения данных модели. Если оперативной памяти недостаточно, то такая модель с большим порядком показывает низкие результаты. Метод PPM особенно эффективен для сжатия текстовых данных.

Предварительные преобразования или фильтрация.

Данные методы служат не для сжатия, а для представления информации в удобном для дальнейшего сжатия виде. Например, для несжатых мультимедиа данных характерны плавные изменения уровня сигнала. Поэтому для них применяют дельта-преобразование, когда вместо абсолютного значения берется относительное. Существуют фильтры для текста, исполняемых файлов, баз данных и другие.

Метод сортировки блока данных (BWT - сокращение от Burrows Wheeler Transform - по имени авторов).

Это особый вид или группа преобразований, в основе которых лежит сортировка. Такому преобразованию можно подвергать почти любые данные. Сортировка производится над блоками, поэтому данные предварительно разбиваются на части. Основным параметром является размер блока, который подвергается сортировке. Для распаковки данных необходимо проделать почти те же действия, что и при упаковке. Поэтому скорость и требования к оперативной памяти почти одинаковы. Архиваторы, которые используют данный метод, обычно показывают высокую скорость и степень сжатия для текстовых данных.

Непрерывные блоки или непрерывный режим (Solid mode - непрерывный режим).

Во многих методах сжатия начальный участок данных или файла кодируется плохо. Например, в словарном методе словарь пуст. В методе контекстного

моделирования модель не построена. Когда количество файлов большое, а их размер маленький, общая степень сжатия значительно ухудшается за счет этих начальных участков. Чтобы этого не происходило при переходе на следующий файл, используется информация, полученная исходя из предыдущих файлов. Аналогичного эффекта можно добиться простым представлением исходных файлов в виде одного непрерывного файла.

Этот метод используется во многих архиваторах и имеет существенный недостаток. Для распаковки произвольного файла необходимо распаковать и файлы, которые оказались в начале архива. Это необходимо для правильного заполнения словаря или построения модели. Существует и промежуточный вариант, когда используются непрерывные блоки фиксированного размера. Потери сжатия получаются минимальными, но для извлечения одного файла, который находится в конце большого архива, необходимо распаковать только один непрерывный блок, а не весь архив.

Сегментирование.

Во всех методах сжатия при изменении типа данных собственно сам переход кодируется очень плохо. Словарь становится не актуальным, модель настроена на другие данные. В этих случаях применяется сегментирование. Это предварительная разбивка на однородные части. Затем эти части кодируются по отдельности или группами.

Особо хочется подчеркнуть, что существует большое количество методов сжатия. Каждый метод обычно ориентирован на один вид или группу реальных данных. Хорошие результаты показывает комплексное использование методов.

Особенности данных

Степень сжатия в основном зависит от исходных данных. Хорошо сжимаются почти все предварительно несжатые данные, например, исполняемые файлы (EXE), тексты (TXT , DOC), базы данных (DBF), простые несжатые изображения (BMP). Ограниченно сжимаются несжатый звук (WAV), сложные несжатые изображения (BMP). Не сжимаются почти все уже сжатые данные, например, архивы (ZIP , CAB), сжатые документы (PDF), сжатая графика и видео (JPG , GIF , AVI , MPG), сжатый звук (MP 3). Их сжатие находится в пределах пары процентов за счет служебных блоков и небольшой избыточности.

Для сжатия некоторых специфических данных (текст, несжатые изображения, несжатый звук) существуют специальные методы и архиваторы. Такие архиваторы обеспечивают высокую степень сжатия и высокую скорость. Однако так называемые универсальные архиваторы постепенно дополняются подобными методами. В данный момент только для несжатого звука существуют высокоэффективные специальные архиваторы, такие, как OptimFROG , Monkey Audio . Для текстов и изображений лучшие универсальные архиваторы показывают лучшую степень сжатия. Например, архив изображений получится меньше, если использовать формат BMP и архиватор WinRK вместо специализированных графических форматов, таких как JPEG 2000 (LossLess - сжатие без потерь).

Большое количество типов данных уже являются сжатыми. Использование архиваторов дает мизерное уменьшение размера. Тем не менее даже в таких случаях эффективное сжатие теоретически возможно. Это обусловлено тем, что в большинстве распространенных форматов файлов, использующих сжатие, применены не самые эффективные методы. Например, в основе формата JPG лежит энтропийное сжатие, которое используется после преобразований Фурье. Данные кодируются неоптимальными блоками, что обусловлено желанием сделать формат JPG устойчивым к повреждениям и возможности частичного извлечения информации. Перекодировав файлы JPG при помощи высокоэффективных методов, можно добиться сжатия порядка 75% от исходного файла (архиватор StuffIt). Собственно сам исходный файл JPG сжимается обычными архиваторами только до 96%. Однако подобные манипуляции с файлами JPG стали возможны только недавно и еще не получили распространения. В большинстве случаев сжимать уже сжатые данные бесполезно.

Следует различать собственно программу-архиватор, формат архивов и методы сжатия. Даже один и тот же метод сжатия может иметь варианты реализации. Например, существует более десятка программ-архиваторов, которые могут создавать архивы в формате ZIP. В свою очередь данные в формате ZIP могут быть сжаты различными методами: Deflate, Deflate64, BZip2. Метод Deflate имеет несколько реализаций с разной скоростью и степенью сжатия (разница порядка 5%). С помощью этого метода архиватор 7-zip позволяет создавать архивы в формате ZIP и 7Z.

Обычно архиваторы могут создавать архивы в собственном эксклюзивном формате с использованием своих оригинальных методов. Например, архиватор RAR позволяет создавать архивы RAR. В формате архива и методах сжатия заключаются основные преимущества того или иного архиватора.

В простейшем случае архиватор позволяет только упаковать или распаковать один файл. Кроме собственно сжатия данных, современные архиваторы обеспечивают некоторые дополнительные функции. Можно выделить несколько основных:

сжатие некоторых файлов и целых директорий;

создание самораспаковывающихся (SFX) архивов. То есть для распаковки архива программа-архиватор не требуется;

изменение содержимого архива;

шифрование содержимого архива;

информация для восстановления архива при частичном повреждении и возможность восстановления поврежденных архивов;

разбивка архива на несколько частей или томов;

консольная версия программы для работы из командной строки;

графическая (GUI) версия программы.

Стоит отметить, что, несмотря на формальное наличие, реализация каждой дополнительной функции может быть выполнена на совершенно разном уровне.

Кроме различий в функциональности, можно разбить архиваторы на две группы: асимметричные и симметричные. Асимметричные архиваторы требуют для операции распаковки значительно меньше времени и оперативной памяти, чем для операции упаковки. Это позволяет быстро получать содержимое архива на маломощных компьютерах. Симметричные архиваторы требуют для операций упаковки и распаковки одинаковое время и объем оперативной памяти. Использование таких архиваторов на широком парке компьютеров или для оперативного доступа к содержимому архива ограничено. Известный архиватор RAR в качестве основного использует асимметричный словарный метод сжатия, а для текстов может использовать симметричный PPM-метод. Таким образом, распаковка архивов RAR, сжатых с максимальной степенью сжатия, может быть невозможна на компьютерах с ограниченным объемом оперативной памяти. Все или почти все передовые архиваторы с высокой степенью сжатия являются симметричными.

Точной статистики по распространенности архиваторов у меня нет. Я выскажу свою субъективную точку зрения на основе личного опыта. Безусловно, самым распространенным архиватором являются ZIP и его модификации. По своей распространенности он значительно превосходит ближайших конкурентов. Следом идут RAR и ACE. В последние годы встречается архиватор 7-zip. Других архиваторов и архивов лично мы не встречали. Исключение составляют некогда популярные ARJ и LHA. В данный момент они не актуальны из-за очень низкой степени сжатия.

Несмотря на очень скромные данные о распространенности архиваторов, их существует большое множество. Основная масса относится к категории экспериментальных и архиваторов с ограниченной функциональностью. Тем не менее, каждый их них позволяет выполнять собственно процедуру сжатия данных. Меньшая распространенность увеличивает вероятность ошибок в программе.

2. Практика.

1. Необходимо создать текстовый файл, содержащий фамилию, имя, отчество студента в объеме 50 записей. Провести архивирование файла. Любым редактором внести изменения согласно задания. В отчете отразить: контрольную сумму исходного файла, сжатого файла, выдаваемые сообщения об ошибках при разархивировании искаженного файла.

2. Провести архивацию файла с паролем. Внести искажения, попробовать разархивировать. В отчете отразить: контрольную сумму исходного файла, сжатого файла, выдаваемые сообщения об ошибках при разархивировании искаженного файла.

3. Провести архивацию файла с паролем, состоящим из 3-х цифр. Провести попытку подбора пароля с использованием программного обеспечения. В отчете отразить: контрольную сумму исходного файла, сжатого файла, выдаваемые сообщения, время подбора.

Варианты:

1. архиватор zip. Искажение двух байт.
2. архиватор arj. Искажение трех байт.
3. архиватор rar. Искажение трех байт.
4. архиватор zip. Удаление двух байт.
5. архиватор arj. Удаление трех байт.
6. архиватор rar. Удаление трех байт.
7. архиватор arj. Добавление трех байт.
8. архиватор rar. Добавление трех байт.
9. архиватор zip. Добавление двух байт.
10. архиватор zip. Удаление двух байт.

Лабораторная работа 3

Шифр простой замены. Таблица Вижинера

1. Теория.

Современные криптографические системы тесно связаны с методами шифрования сообщений, которые, в свою очередь, зависят от способа использования ключей. Предлагаемая программа отличается простотой понимания смысла шифрования, позволяет получить криптограммы одного и того же исходного текста в зависимости от выбранного ключевого слова. Кроме того, такой подход шифрования может быть применен в одноключевых криптосистемах для защиты информации в локальных сетях.

Одноключевые криптографические системы являются классическими системами криптографической защиты информации. Для шифрования и расшифрования сообщений в них используется один и тот же ключ, сохранение которого в тайне обеспечивает надежность защиты информации. Шифровальную схему в этом случае можно представить следующим образом:

$$Y = E_z(X)$$
$$X = D_z(Y) = D_z(E_z(X)),$$

где X — открытый текст;

Y — шифротекст;

D_z — функция шифрования с секретным ключом z;

E_z — функция расшифрования с секретным ключом z.

Открытый текст, как правило, имеет произвольную длину. В связи с этим он разбивается на блоки фиксированной длины и каждый блок шифруется в отдельности, независимо от его получения во входной последовательности. Соответствующие методы шифрования называются блочными, а наиболее важными шифрами при этом являются шифры замены (подстановки). Шифры замены образуются с помощью замены знаков исходного сообщения на другие знаки.

Простейшим шифром замены является шифр Цезаря. В этом шифре буквы исходного сообщения латинского алфавита заменяются буквами, расположенными тремя позициями правее. Однако вскрытие таких шифров легко осуществляется путем перебора всех возможных ключей, в качестве которых используется величина сдвига букв сообщения в алфавите, до появления осмысленного текста.

Устойчивость шифра замены можно повысить за счет использования «перемешанного» алфавита. Однако наиболее стойким к расшифрованию сообщений из данного класса шифров является шифр полиалфавитной замены, в котором применяется несколько алфавитов, поочередно используемых для замены букв открытого текста.

Разновидностью шифрования с использованием полиалфавитной замены знаков сообщения является метод Вижинера (или шифр Вижинера), в котором важную роль играет ключевое слово.

Приведем в качестве примера программу шифрования текста сообщения с помощью шифра Вижинера. Программа может быть применена для создания шифротекстов с последующей передачей их в одноключевых криптосистемах.

Математическая постановка такой задачи заключается в следующем. Множество из 26 алфавитов, для английского текста (по числу букв), формируется последовательным циклическим сдвигом букв исходного алфавита (аналогично принципу формирования шифра Цезаря). Совокупность всех алфавитов образует так называемую таблицу Вижинера.

При шифровании буквы ключевого слова определяют выбор конкретного сдвинутого алфавита, используемого при замене соответствующей буквы сообщения.

Процесс шифрования может быть описан как процесс суммирования по модулю 26 номеров соответствующих друг другу букв открытого текста и ключевого слова.

В данном случае для уяснения принципа получения криптограмм с использованием шифра Вижинера применим ключевое слово и один алфавит английского языка.

Каждой букве алфавита сопоставим цифру (A $^{\Omega\Omega}$ 0, B $^{\Omega\Omega}$ 1, …, Z $^{\Omega\Omega}$ 25). Ключевое слово k_i задается определенным количеством букв d и повторно записывается под шифруемым сообщением m_i. В дальнейшем в i-м столбце из двух букв буква сообщения m_i складывается по модулю 26 со стоящей под ней буквой ключевого слова k_i в виде:

$$g_i = m_i + k_i \bmod 26,$$

где g_i — буквы полученной криптограммы.

Расшифровка криптограммы осуществляется вычитанием ключевого слова по модулю 26. При $d = 1$ шифр Вижинера является шифром Цезаря.

2. Примеры.

Пример 1. Шифр Цезаря

Получим криптограмму с использованием шифра Цезаря с ключом $d = 1$ на базе английского алфавита, строчный регистр.

Исходное сообщение	i	n	f	o	r	m	f	t	i	o	n
Криптограмма	j	o	g	p	s	n	g	u	j	p	o

Для русского языка с ключом $d = 10$

Исходное сообщение	И	Н	Ф	О	Р	М	А	Ц	И	Я
Криптограмма	Т	Ч	Ю	Ш	Ь	Ц	Й	А	Т	И

Пример 2. Шифр Вижинера

Получим криптограмму с использованием шифра Цезаря с ключевым словом «code» (ключи «c=2», «o=14», «d=3», «e=4») на базе английского алфавита, строчный регистр.

Исходное сообщение	i	n	f	o	r	m	f	t	i	o	n
Ключевое слово	c	o	d	e	c	o	d	e	c	o	d
Криптограмма	k	b	i	s	t	a	i	x	k	c	q

Для русского языка с ключевым словом «код» (ключи «к=10», «о=14», «д=4»).

Исходное сообщение	И	Н	Ф	О	Р	М	А	Ц	И	Я
Ключевое слово	К	О	Д	К	О	Д	К	О	Д	К
Криптограмма	Т	Ы	Ш	Ш	Ю	Р	Й	Д	М	И

3. Практика.

Составьте алгоритмическое и программное обеспечение:

1. Процедур шифрования и расшифрования с использованием шифра Цезаря при вводе с клавиатуры ключа и исходного или зашифрованного текста. Учтите регистр вводимого текста.

2. Процедур шифрования и расшифрования с использованием шифра Цезаря при вводе с клавиатуры ключа и текстового файла. Учтите регистр вводимого текста.
3. Процедур шифрования и расшифрования с использованием шифра Вижинера при вводе с клавиатуры ключа и исходного или зашифрованного текста. Учтите регистр вводимого текста.
4. Процедур шифрования и расшифрования с использованием шифра Вижинера при вводе с клавиатуры ключа и текстового файла. Учтите регистр вводимого текста.
5. Постройте программно таблицу Вижинера и выведите в файл.

Для созданного программного обеспечения проведите тестирование не менее чем на 10 различных наборах данных.

Лабораторная работа 4

Обмен ключами по Диффи-Хелману

1. Теория.

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

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

- Одноключевые (симметричные)
- Двухключевые (несимметричные)

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

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

Отмеченный парадокс может быть исключен использованием идеи Диффи и Хеллмана, которые предложили способ выработки секретного ключа без предварительного согласования между абонентами сети путем обмена информацией по открытому каналу. Этот способ был предложен Диффи и Хеллманом в 1976 году и опубликован в ряде работ по криптографии. Реализация такого способа привела к появлению открытого шифрования. Абонент сего открыто сообщал о том, каким образом зашифровать к нему сообщение, расшифровать же его мог только он сам.

Основную роль при выработке секретного ключа в данном случае играют математические операции, когда прямая операция сравнительно проста, а обратная — практически трудно реализуема.

Прямая операция: возвести основание a в степень p и взять остаток по модулю m вида:

$L = a^p \bmod m$.

Обратная операция: найти p, зная L, a и m.

Обратная операция (задача) при этом может быть решена простым перебором значений p, но практически не решается при больших значениях p .

В предлагаемом алгоритме выработки секретного ключа известны основание a и mod m.

Отправитель сообщения с помощью генератора случайных чисел (ГСЧ) получает случайное число X ($1 < x < m$), вычисляет значение $L_0 = a^y \bmod m$ и посылает L_0 получателю.

Получатель принимает L_0 вырабатывает с помощью своего ГСЧ случайное число Y ($1 < y < m$), вычисляет значение $L_p = a^y \bmod m$ и посылает L_p отправителю.

Отправитель принимает L_p, вычисляет $K_0 = L_p^{\,x} = a^{yx} \bmod m$. Получатель вычисляет $K_p = L_0^{\,y} = a^{yx} \bmod m$.

Так как $K_0 = K_p$., то это число и является общим секретным ключом.

Злоумышленник, перехватив L_0 и L_p , не знает случайных чисел x и y и не сможет расшифровать исходный текст сообщения.

3. Практика.

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

Лабораторная работа 5

Шифр RSA

1. Теория.

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

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

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

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

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

Функционирование криптосистемы на основе метода RSA предполагает формирование открытого и секретного ключей. С этой целью необходимо выполнить следующие математические операции:

- Выбираем два больших простых числа p и q, понимая под простыми числами такие числа, которые делятся на само себя и число 1
- Определяем $n = p \times q$,
- Вычисляем число $k = (p-1)\,(q-1)$,
- Выбираем большое случайное число d, взаимно простое с числом k (взаимно простое число — это число, которое не имеет ни одного общего делителя, кроме числа 1)
- Определяем число e, для которого истинным является соотношение

$$(e \times d) \bmod k = 1$$

- Принимаем в качестве открытого ключа пару чисел $\{e, n\}$
- Формирование секретного ключа в виде пары чисел $\{d, n\}$

Для зашифровки передаваемых данных с помощью открытого ключа $\{e, n\}$ необходимо выполнить операции:

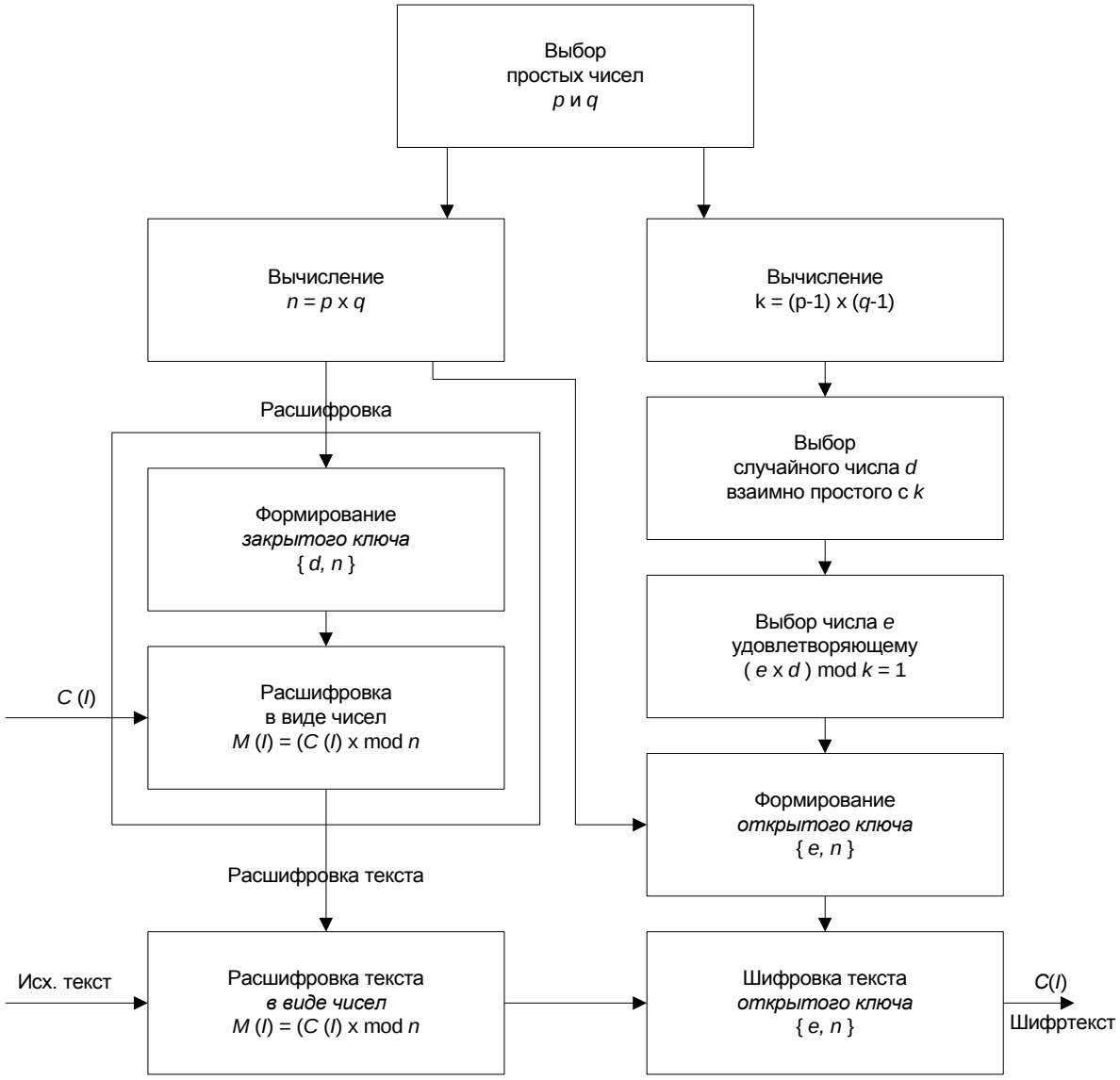
- Разбить шифруемый текст на блоки, каждый из которых может быть представлен в виде чисел $M(i) = 0,1,..., n-1$
- Зашифровать текст в виде последовательности чисел $M(i)$ по формуле

$$C(i) = (M(i)^e) \bmod n$$

- Расшифрование шифротекста производится с помощью секретного ключа $\{d, n\}$ при выполнении следующих вычислений:

$$M(i) = (C(i)^d) \bmod n$$

В результате получаем последовательность чисел $M(i)$, представляющих исходные данные. На практике при использовании метода RSA длина p и q составляет 100 и более десятичных знаков, что обеспечивает высокую криптостойкость шифротекста.



3. Практика.

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

Лабораторная работа 6

Циклические коды

1. Теория.

Принцип работы – метод проверки целостности массива бит, основанный на свойствах операции взятия остатка в полиномиальной арифметике по модулю 2 с основными операциями 0+0=0, 0+1=1, 1+0=1, 1+1=0, 0*0=0, 0*1=0, 1*0=0, 1*1=1.

CRC Cyclic Redundancy Check - (Циклический избыточный контрольный код) результат операции взятия остатка от деления проверяемого битового массива на некоторое число-делитель, которое обладает специфическими свойствами равномерно "размазывать" изменение в некотором бите массива на возможно большее число бит результата. Это число-делитель, называемое образующим полиномом, выбирается так, чтобы само являлось полиномиально простым - не делилось полиномиально нацело на любые числа от 2 до самого себя. кроме того, есть и другие критерии выбора полинома, направленные на уменьшение вероятности пропуска типичных ошибок в каналах передачи данных, так что самостийно выдумывать полиномы - дело не только трудное, но и вредное.

Полином может быть записан как в виде суммы степеней с ненулевыми (а значит - единичными) коэффициентами, так и маской этих единичек. Порядок записи единиц в маске однозначно связан с порядком обработки бит в проверяемом массиве, потому что в процессе расчета CRC промежуточный результат необходимо циклически сдвигать в ту же сторону, что и биты проверяемого массива, причем сдвигать так, чтобы вытеснялись старшие степени полинома. Самая старшая степень в маске не учитывается, она определяет только число бит маски. Ниже старшие степени отделены пропусками. Чтобы реализовать проверку с применением CRC, помимо маски полинома и порядка следования бит в массиве (определяющего направление циклического сдвига), необходимо знать начальное значение CRC и метод завершающей модификации результата вычисления CRC.

Типичные методы, применяемые для контроля целостности данных при передаче и хранении:

CCITT-CRC-32 [Все распространенные архиваторы и протоколы с CRC-32] биты массива обрабатываются, начиная с младшего бита в байте - LSB. Образующий полином: X^0+X^1+X^2+X^4+X^5+X^7+X^8+X^10+X^11+X^12+X^16+X^22+X^23+X^ 26 +X^32 Маска = EDB88320h, в которой правые цифры соответствуют старшим степеням, сдвиг выполняется вправо. Начальное значение - 0xFFFFFFFF. Конечная модификация - поразрядная инверсия всех битов результата.

CCITT-DOS-16 [архиватор LHA и, вероятно, некоторые другие с CRC-16] биты массива обрабатываются, начиная с младшего бита в байте - LSB. Образующий полином: X^0+X^2+X^15 +X^16 Маска = A001h, в которой правые цифры соответствуют старшим степеням, сдвиг выполняется вправо. Начальное значение - 0000. Конечная модификация - отсутствует.

CCITT-CRC-16 [протоколы передачи данных с CRC-16, Контроль EMSI] биты массива обрабатываются, начиная со старшего бита в байте - MSB. Образующий полином: X^16+ X^12+X^5+X^0 Маска = 0x1021, в которой левые цифры соответствуют старшим степеням, сдвиг выполняется влево. Начальное значение - 0x0000. Конечная модификация - отсутствует.

Теперь собственно описание вычисления: Рабочая переменная W соответствующей разрядности, в которой будет накапливаться результат, инициализируется начальным значением. Затем для каждого бита *m* входного массива выполняются следующие действия: *W* сдвигается на 1 бит (о направлении сдвига см. выше) В освободившийся бит

W помещается нуль. Бит, только что вытолкнутый из *W*, сравнивается с битом *m*. если они не совпали, выполняется операция исключающего ИЛИ над *W* и маской полинома, результат заносится в *W*. И так, пока не будут обработаны все биты массива. После чего над *W* производится конечная модификация.

Можно сказать, что обычно так CRC считают только в схемных реализациях. Потому, что это очень медленно - ведь число циклов равно числу бит массива. При реализации на программном уровне обработка ведется восьмерками бит - байтами. Заводится табличка из 256 элементов. Каждое значение - результат расчета CRC над восьмеркой бит индекса элемента: for i := 0 to 255 do tab[i] := count_crc(*i*); После этого расчет CRC для массива можно вести байтами. Начало и конец расчета, как и раньше. А цикл идет для каждого байта *Q*: *W* := *W* XOR *Q*; *W* := сдвиг(*W*, 8) XOR tab[*W*]

При LSB-порядке Q операция XOR выполняется над младшими битами W, а при MSB-порядке - над старшими. Индексом в таблице служат именно эти биты.

Байтовый табличный метод требует ощутимых затрат памяти под таблицу. Для CRC-32 требуется таблица размером в килобайт. Если килобайт памяти для реализации с одним циклом на байт массива это перебор, можно предложить компромиссный вариант - считать CRC, не восьмерками, а четверками бит. CRC-32-таблица из 16 значений займет 64 байта, но скорость будет несколько ниже, чем при большой таблице, хотя существенно выше, чем без нее вообще.

В реализации расчета CRC для z-modem'a есть один тонкий момент. Там допущено отклонение от базовой схемы. Две строки поменяли местами: *W* := сдвиг(*W*, 8) XOR tab[*W*]; *W* := *W* XOR *Q*

В результате получается не чистый CRC: контрольный код z-modem'a для массивов до двух байт размером "равен" самому массиву. Например, для массива из двух байт 3 и 8, контрольный код будет равен 0308h.

[И наконец, одно маленькое замечание. Операция вычисления CRC обратима. Не в том смысле, конечно, что по CRC можно восстановить весь массив, а в том, что если дано CRC разрядности *N* и дан некоторый массив, в котором где-нибудь можно поменять подряд *N* бит, то подогнать этот массив под заданную CRC не сложнее, чем посчитать CRC. CRC не является криптографически устойчивой хеш-функцией.]

3. Практика.

Составьте алгоритмическое и программное обеспечение, реализующее алгоритм CRC. В качестве исходных данные – файл. Для созданного программного обеспечения проведите тестирование не менее чем на 10 различных наборах данных.