СОДЕРЖАНИЕ

Введение

В постановлении Президента Республики Узбекистан «О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий» от 21 марта 2012 год, № ПП-1730 [1] особое внимание уделено вопросам «Совершенствования системы регулирования в сфере информационно-коммуникационных технологий с учетом состояния развития информационных ресурсов технологий и систем...».

Интенсивное внедрение информационных систем в различные сферы деятельности человека позволяет в сотни и тысячи раз повысить оперативность и эффективность решения задач информационного процесса, автоматизации процессов управления производством, сбора и обработки статистических данных. Объемы передаваемой и обрабатываемой информации возросли многократно, причем информационные технологии начали сливаться с телекоммуникациями. Этот процесс происходит и в нашей республике: расширяется сеть передачи данных, вводятся в эксплуатацию новые цифровые системы, расширяется сеть мобильной связи и т.д. Таким образом, быстро развивающиеся компьютерные информационные технологии вносят заметные изменения в нашу жизнь. Все чаще понятие «информация» используется как обозначение специального товара, который можно приобрести, продать, обменять на что то другое. При этом стоимость информации в сотни и тысячи раз превосходит стоимость компьютерных систем, в которых она находится. Поэтому вполне естественно возникает необходимость в защите информации от несанкционированного доступа, умышленного изменения, кражи, уничтожения и других преступных действий.

Криптографические методы представляют собой способ защиты информации путем приведения ее к неявному виду, преобразовывая информацию с помощью специальных алгоритмов либо аппаратных средств и кодов ключей.

Основным задачам, решающе криптографическими методами относится:

- защита передаваемых и хранимых секретных данных от разглашения и искажения. Одна из основных задач, имеющих наиболее широкое распространение;

- задача подтверждения авторства сообщения (электронная цифровая подпись);

- вручение сообщения под расписку. Здесь имеется в виду взаимная аутентификация, как источника формирующего сообщение, так и получателя кому предназначалось сообщение.

Таким образом, криптографические методы используются не только для скрытия информации, но и в алгоритмах аутентификации.

Целью настоящего выпускной квалификационной работы является исследованию методов обеспечение целостности информации в сетях передачи данных.

Исходя из вышесказанного можно сделать вывод об актуальности выбранной темы выпускной работы.

Выпускная квалификационная работа состоит из введения, трёх разделов, заключения, списка используемой литературы и приложения.

Во введении обоснована актуальность темы выпускной квалификационной работы.

В первом разделе рассматриваются общие понятия об угрозы в компьютерных сетях и методы обеспечение целостности информации. Отражается интегрированный подход, включающий в себя девять абстрактных теоретических принципов, обеспечивающие целостность данных в информационных системах. Исследуется модель контроля целостности данных, определяющая категории объектов данных и два класса операций над ними. Анализируются правил, определяющие взаимоотношения элементов данных и процедур в процессе

функционирования системы. А также описываются особенности обеспечение целостности информации в компьютерных сетях.

Во втором разделе анализируются избыточные и CRC коды, позволяющей обнаруживать и исправлять одиночные ошибки целостности информации. Формализуется алгоритм расчёта CRC16 для получения контрольной суммы полиномов. Анализируются алгоритмы хэш-функции и их функциональные роли в обеспечения целостности информации в сетях передачи данных. Исследуются методов обеспечение целостностью информации с помощью HMAC. Предлагается программа, позволяющая шифровать и дешифровать информации на основе алгоритм RC6.

В третьем разделе рассматривается безопасность жизнедеятельности.

В заключении приведены основные выводы выпускной квалификационной работы.

1. Теоретическая часть. Анализ целостности информации в сетях передачи данных

1.1. Классификации основных угроз защиты информации в сетях передачи данных

Информатизация является характерной чертой жизни современного общества. Новые информационные технологии активно внедряются во все сферы народного хозяйства. Компьютеры управляют космическими кораблями и самолетами, контролируют работу атомных электростанций, распределяют электроэнергию и обслуживают банковские системы. Компьютеры являются основой множества автоматизированных систем обработки информации (АСОИ), осуществляющих хранение и обработку информации, предоставление ее потребителям, реализуя тем самым современные информационные технологии. По мере развития и усложнения средств, методов и форм автоматизации процессов обработки информации повышается зависимость общества от степени безопасности используемых им информационных технологий, от которых порой зависит благополучие, а иногда и жизнь многих людей. Актуальность и важность проблемы обеспечения безопасности информационных технологий обусловлены следующими причинами:

- резкое увеличение вычислительной мощности современных компьютеров при одновременном упрощении их эксплуатации;
- резкое увеличение объемов информации, накапливаемой, хранимой и обрабатываемой с помощью компьютеров и других средств автоматизации;
- сосредоточение в единых базах данных информации различного назначения и различной принадлежности;
- высокие темпы роста парка персональных компьютеров, находящихся в эксплуатации в самых разных сферах деятельности;
- резкое расширение круга пользователей, имеющих непосредственный доступ к вычислительным ресурсам и массивам данных;

- бурное развитие программных средств, не удовлетворяющих даже минимальным требованиям безопасности;
- повсеместное распространение сетевых технологий и объединение локальных сетей в глобальные;
- развитие глобальной сети Internet, практически не препятствующей нарушениям безопасности систем обработки информации во всем мире.

Безопасность АСОИ достигается принятием мер по обеспечению конфиденциальности и целостности обрабатываемой ею информации, а также доступности и целостности компонентов и ресурсов системы.

Конфиденциальность данных - это статус, предоставленный данным и определяющий требуемую степень их защиты. По существу конфиденциальность информации- это свойство информации быть известной только допущенным и прошедшим проверку (авторизированным) субъектам системы (пользователям, процессам, программам). Для остальных субъектов системы эта информация должна быть неизвестной[2].

Целостность информации обеспечивается в том случае, если данные в системе не отличаются в семантическом отношении от данных в исходных документах, т.е. если не произошло их случайного или преднамеренного искажения или разрушения.

Целостность компонента или ресурса системы - это свойство компонента или ресурса быть неизменными в семантическом смысле при функционировании системы в условиях случайных или преднамеренных искажений или разрушающих воздействий.

Доступность компонента или ресурса системы - это свойство компонента или ресурса быть доступным для авторизованных законных субъектов системы.

По цели воздействия различают три основных типа угроз безопасности АСОИ:

- угрозы нарушения конфиденциальности информации;
- угрозы нарушения целостности информации;

- угрозы нарушения работоспособности системы (отказы в обслуживании).

Угрозы нарушения конфиденциальности направлены на разглашение конфиденциальной или секретной информации. При реализации этих угроз информация становится известной лицам, которые не должны иметь к ней доступ. В терминах компьютерной безопасности угроза нарушения конфиденциальности имеет место всякий раз, когда получен несанкционированный доступ к некоторой закрытой информации, хранящейся в компьютерной системе или передаваемой от одной системы к другой.

Угрозы нарушения целостности информации, хранящейся в компьютерной системе или передаваемой по каналу связи, направлены на ее изменение или искажение, приводящее к нарушению ее качества или полному уничтожению. Целостность информации может быть нарушена умышленно злоумышленником, а также в результате объективных воздействий со стороны среды, окружающей систему. Эта угроза особенно актуальна для систем передачи информации - компьютерных сетей и систем телекоммуникаций. Умышленные нарушения целостности информации не следует путать с ее санкционированным изменением, которое выполняется полномочными лицами с обоснованной целью (например, таким изменением является периодическая коррекция некоторой базы данных).

Угрозы нарушения работоспособности (отказ в обслуживании) направлены на создание таких ситуаций, когда определенные преднамеренные действия либо снижают работоспособность АСОИ, либо блокируют доступ к некоторым ее ресурсам. Например, если один пользователь системы запрашивает доступ к некоторой службе, а другой предпринимает действия по блокированию этого доступа, то первый пользователь получает отказ в обслуживании. Блокирование доступа к ресурсу может быть постоянным или временным.

Нарушения конфиденциальности и целостности информации, а также доступности и целостности определенных компонентов и ресурсов АСОИ могут быть вызваны различными опасными воздействиями на АСОИ.

Несанкционированный доступ (НСД) является наиболее распространенным и многообразным видом компьютерных нарушений. Суть НСД состоит в получении пользователем (нарушителем) доступа к объекту в нарушение правил разграничения доступа, установленных в соответствии с принятой в организации политикой безопасности. НСД использует любую ошибку в системе защиты и возможен при нерациональном выборе средств защиты, их некорректной установке и настройке. НСД может быть осуществлен как штатными средствами АСОИ, так и специально созданными аппаратными и программными средствами.

Перечисляются основные каналы несанкционированного доступа, через которые нарушитель может получить доступ к компонентам АСОИ и осуществить хищение, модификацию и/или разрушение информации:

- все штатные каналы доступа к информации (терминалы пользователей, оператора, администратора системы; средства отображения и документирования информации; каналы связи) при их использовании нарушителями, а также законными пользователями вне пределов их полномочий;
- технологические пульты управления;
- линии связи между аппаратными средствами АСОИ;
- побочные электромагнитные излучения от аппаратуры, линий связи, сетей электропитания и заземления и др.

Из всего разнообразия способов и приемов несанкционированного доступа остановимся на следующих распространенных и связанных между собой нарушениях:

- перехват паролей;
- «маскарад»;
- незаконное использование привилегий.

Перехват паролей осуществляется специально разработанными программами. При попытке законного пользователя войти в систему программа-перехватчик имитирует на экране дисплея ввод имени и пароля пользователя, которые сразу пересылаются владельцу программы-перехватчика, после чего на экран выводится сообщение об ошибке и управление возвращается операционной системе. Пользователь предполагает, что допустил ошибку при вводе пароля. Он повторяет ввод и получает доступ в систему. Владелец программы-перехватчика, получивший имя и пароль законного пользователя, может теперь использовать их в своих целях. Существуют и другие способы перехвата паролей.

«Маскарад»-это выполнение каких-либо действий одним пользователем от имени другого пользователя, обладающего соответствующими полномочиями. Целью «маскарада» является приписывание каких-либо действий другому пользователю либо присвоение полномочий и привилегий другого пользователя.

Примерами реализации «маскарада» являются:

- вход в систему под именем и паролем другого пользователя (этому «маскараду» предшествует перехват пароля);
- передача сообщений в сети от имени другого пользователя.

«Маскарад» особенно опасен в банковских системах электронных платежей, где неправильная идентификация клиента из-за «маскарада» злоумышленника может привести к большим убыткам клиента банка.

Незаконное использование привилегий. Большинство систем защиты устанавливают определенные наборы привилегий для выполнения заданных функций. Каждый пользователь получает свой набор привилегий: обычные пользователи - минимальный, администраторы - максимальный.

Несанкционированный захват привилегий, например посредством «маскарада», приводит к возможности выполнения нарушителем определенных действий в обход системы защиты. Следует отметить, что незаконный захват привилегий возможен либо при наличии ошибок в

системе защиты, либо из-за халатности администратора при управлении системой и назначении привилегий.

Особо следует остановиться на угрозах, которым могут подвергаться компьютерные сети. Основная особенность любой компьютерной сети состоит в том, что ее компоненты распределены в пространстве. Связь между узлами (объектами) сети осуществляется физически с помощью сетевых линий связи и программно с помощью механизма сообщений. При этом управляющие сообщения и данные, пересылаемые между объектами сети, передаются в виде пакетов обмена. При вторжении в компьютерную сеть злоумышленник может использовать как пассивные, так и активные методы вторжения.

При пассивном вторжении (перехвате информации) нарушитель только наблюдает за прохождением информации по каналу связи, не вторгаясь ни в информационный поток, ни в содержание передаваемой информации. Как правило, злоумышленник может определить пункты назначения и идентификаторы либо только факт прохождения сообщения, его длину и частоту обмена, если содержимое сообщения не распознаваемо, т.е. выполнить анализ трафика (потока сообщений) в данном канале.

При активном вторжении нарушитель стремится подменить информацию, передаваемую в сообщении. Он может выборочно модифицировать, изменить или добавить правильное или ложное сообщение, удалить, задержать или изменить порядок следования сообщений. Злоумышленник может также аннулировать и задержать все сообщения, передаваемые по каналу[3]. Компьютерные сети характерны тем, что кроме обычных локальных атак, осуществляемых в пределах одной системы, против объектов сетей предпринимают так называемые удаленные атаки, что обусловлено распределенностью сетевых ресурсов и информации. Злоумышленник может находиться за тысячи километров от атакуемого объекта, при этом нападению может подвергаться не только конкретный компьютер, но и информация, передающаяся по сетевым каналам связи. Под

удаленной атакой понимают информационное разрушающее воздействие на распределенную компьютерную сеть, программно осуществленное по каналам связи.

В табл.1.1 показаны основные пути реализации угроз безопасности АСОИ при воздействии на ее компоненты. Конечно, табл.1.1 дает самую общую картину того, что может произойти с системой.

Таблица 1.1

Пути реализации угроз безопасности АСОИ

Объекты воздействия	Нарушение конфиденциальности информации	Нарушение целостности информации	Нарушение работоспособности системы
Аппаратные средства	НСД - подключение; использование ресурсов; хищение носителей	НСД - подключение; использование ресурсов; модификация, изменение режимов	НСД – изменение режимов; вывод из строя; разрушение
Программное обеспечение	НСД - копирование; хищение; перехват	НСД, внедрение «троянского коня», «вирусов», «червей»	НСД – искажение; удаление; подмена
Данные	НСД - копирование; хищение; перехват	НСД - искажение; модификация	НСД - искажение; удаление; подмена
Персонал	Разглашение; передача сведений о защите; халатность	«Маскарад»; вербовка; подкуп персонала	Уход с рабочего места; физическое устранение

Для защиты от указанных вредоносных программ необходимо применение ряда мер:

- исключение несанкционированного доступа к исполняемым файлам;
- тестирование приобретаемых программных средств;
- контроль целостности исполняемых файлов и системных областей;
- создание замкнутой среды исполнения программ.

1.2. Механизмы контроля целостности данных

Целостность информации – это способность средства вычислительной техники или автоматизированной системы обеспечивать неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения).

Под *угрозой нарушения целостности* понимается любое умышленное изменение информации, хранящейся в вычислительной системе или передаваемой из одной системы в другую. Когда злоумышленники преднамеренно изменяют информацию, говорится, что целостность информации нарушена. Целостность также будет нарушена, если к несанкционированному изменению приводит случайная ошибка программного или аппаратного обеспечения[4].

Метод контрольных сумм. Наиболее простым и одним из самых первых методов обеспечения целостности данных является метод *контрольных сумм*. Под контрольной суммой понимается некоторое значение, полученное путем сложения всех чисел входных данных в конечном множестве.

Пусть:

a – массив данных, элементами которого являются числовые значения.

Length(**a**) – количество элементов массива **a**.

Sum – результат суммирования всех элементов **a**[i] массива данных **a**, где i = [1..*Length*(**a**)].

CheckSum – контрольная сумма массива **a**.

MaxVal – максимально возможное числовое значение *CheckSum*.

Тогда контрольной суммой массива **a** будет являться величина *CheckSum* полученная путем деления с остатком суммы всех элементов массива *Sum* на максимально возможное числовое значение контрольной суммы, увеличенное на единицу или:

$$CheckSum = Sum \bmod (MaxVal+1)$$

Рассмотрено вышеописанный механизм расчета контрольной суммы на примере.

Пусть есть некая последовательность байт:

a = 0A 11 B3 58 A2 CE 78 6D 90 01

Считано, что под контрольную сумму *CheckSum* нам выделяется ячейка памяти размерностью 1 байт (максимальное значение *MaxVal* = FF или 255 в десятичной системе исчисления)

Рассчитается сумму байт последовательности:

Sum(a[i]) = 0A+11+B3+58+A2+CE+78+6D+90+01=40C(или 1036 в десятичной системе исчисления), где i = [1..10].

Контрольной суммой массива **a** будет являться величина *CheckSum* полученная путем деления с остатком суммы всех элементов массива *Sum* на максимально возможное числовое значение контрольной суммы, увеличенное на единицу.

CheckSum = *Sum* **mod** (*MaxVal*+1) = 40C **mod** (FF+1) = 1036 **mod** (255+1) = 12 (0C в шестнадцатеричной системе исчисления)

Таким образом, получено, что контрольной суммой массива байт **a** является числовое значение 12. Стоит заметить, что при изменении какого-либо элемента массива a изменится и его контрольная сумма.

Использование метода контрольных сумм началось еще на заре вычислительной техники, и он применяется до сих пор в некоторых протоколах передачи данных.

Рассмотрены плюсы и минусы метода:

Плюсы:

- простота реализации;
- высокая производительность.

Минусы:

- равенство полученных значений контрольных сумм не дает гарантии неизменности информации;

- при известной контрольной сумме крайне просто «подогнать» данные так, чтобы в результате получилось корректное значение.

Подытожив, заметано, что хотя использования контрольных сумм для верификации целостности данных на этапе зарождения компьютерных технологий было достаточно, но на сегодняшний день, данный метод является атавизмом.

Метод «циклического контрольного кода». Более совершенным способом контроля целостности данных является, так называемый, метод «*циклического контрольного кода*»(cyclic redundancy check - CRC). Алгоритм широко используется в аппаратных устройствах (дисковые контроллеры, сетевые адаптеры и др.) для верификации неизменности входной и выходной информации, а также во многих программных продуктах для выявления ошибок при передаче данных по каналам связи. В основе метода CRC лежит понятие *полинома* или *многочлена*. Каждый бит некоторого блока данных соответствует одному из коэффициентов двоичного полинома. Например, полином шестнадцатеричного числа 7A (двоичная запись - 1111010) будет выглядеть следующим образом:

$A(x) = 1*x^6+1*x^5+1*x^4+1*x^3+0*x^2+1*x^1+0*x^0 = x^6+x^5+x^4+x^3+x$

Таким образом, любой блок данных представляет собой последовательность битов, которую можно представить в виде двоичного полинома A(x). Для вычисления контрольного кода необходим еще один полином G(x), называемый *порождающим* полиномом. Для каждой реализации алгоритма контроля CRC порождающий полином выбирается заранее произвольным образом. Например, для контроллеров гибких магнитных дисков порождающий полином $G(x) = x^{16}+x^{12}+x^5+1$.

Пусть R(x) - некий полином. R(x) называется контрольным кодом полинома A(x) при порождающем полиноме G(x), если R(x) является остатком от деления полинома $A(x)*x^r$ на G(x), где r – степень полинома G(x).

$R(x) = (A(x)*x^r) \bmod G(x)$

Так же, как и для контрольных сумм, контрольный код не занимает много места (обычно 16/32 бита), однако вероятность обнаружения ошибки существенно выше. Например, в отличие от контрольных сумм метод CRC сможет обнаружить перестановку двух байт либо добавление единицы к одному и вычитание единицы из другого.

Рассмотрены плюсы и минусы метода:

Плюсы:

- простота реализации;
- высокая производительность;
- высокая вероятность обнаружения ошибки.

Минусы:

- равенство полученных значений контрольных сумм все равно не дает гарантии неизменности информации;
- при известной контрольной сумме не составляет большого труда написать программу, которая будет «подгонять» данные так, чтобы в результате получилось корректное значение, хотя это намного сложнее, чем в предыдущем методе контрольных сумм.

Подытожив, можно сказать, что метод «циклического контрольного кода» превосходно обнаруживает случайные изменения в данных, однако недостаточно надежен в случае несанкционированного изменения информации злоумышленником.

Однонаправленные функции хэширования. Существенно более высокой надежности, чем при методе «циклического контрольного кода», можно достичь, используя однонаправленные функции «хэширования». Термин «однонаправленный» означает следующее:

Пусть имеется некая функция *f* и произвольный набор данных A.

Пусть результатом применения функции *f* к A является набор данных B (хэш).

f(A) = B

Функция f является однонаправленной, если не существует такой функции g, что

g(B) = A

либо такую функцию g крайне сложно построить.

Коллизией называется ситуация, когда разным наборам входных блоков данных соответствует один хэш. Важная отличительная особенность хороших алгоритмов хэширования заключается в том, что генерируемые с его помощью значения настолько уникальны и трудно повторимы, что задача нахождения коллизий является чрезвычайно тяжелой как по ресурсоемкости, так и по производительности[5]. Вышесказанное можно записать следующим образом:

Пусть *f*(A)=B. Не существует такого A’ либо его крайне сложно найти, что *f*(A’)=B. Чем больше длина хэша, тем труднее найти соответствующий набор входных данных.

Среди алгоритмов хэширования наибольшей известностью пользуются:

- алгоритм MD5 (длина хэша – 128 бит), автор Ron Rivest, создатель алгоритма шифрования с открытым ключом RSA
- алгоритм SHA-1 (длина хэша – 160 бит), созданный усилиями специалистов Национального института по стандартизации и технологиям (NIST) и Агентства национальной безопасности (NSA).

1.3. Принципы, обеспечивающие целостность данных в сетях передачи данных

При рассмотрении вопроса целостности данных используется интегрированный подход (в определенном выше смысле), а также их последователей и оппонентов, и включающий в себя девять абстрактных теоретических принципов, каждый из которых раскрывается ниже:

- корректность транзакций;
- аутентификация пользователей;

- минимизация привилегий;
- разграничение функциональных обязанностей;
- аудит произошедших событий;
- объективный контроль;
- управление передачей привилегий;
- обеспечение непрерывной работоспособности;
- простота использования защитных механизмов.

Пользователь не должен модифицировать данные произвольно, а только определенными способами, так, чтобы сохранялась целостность данных. Другими словами, данные можно изменять только путем корректных транзакций и нельзя произвольными средствами. Кроме того, предполагается, что «корректность» (в обычном смысле) каждой из таких транзакций может быть некоторым способом доказана. Принцип корректных транзакций по своей сути отражает основную идею определения целостности данных сформулированную выше.

Второй принцип гласит, что изменение данных может осуществляться только специально аутентифицированными для этой цели пользователями. Этот принцип работает совместно с последующими четырьмя, с которыми тесно связана его роль в общей схеме обеспечения целостности. Идея минимизации привилегий появилась еще на ранних этапах развития направления компьютерной безопасности в форме ограничения, накладываемого на возможности процессов, выполняющихся в АС, и подразумевающего, что процессы должны быть наделены теми и только теми привилегиями, которые естественно и минимально необходимы для выполнения процессов. Практикам администрирования ОС UNIX это положение хорошо знакомо на примере правил использования учетной записи rot, обладающей неограниченными полномочиями. Принцип, согласно которому следует минимизировать назначаемые привилегии в строгом соответствии с содержанием выполняемой задачи, распространяется

в равной мере как на процессы (работающие в системе программы), так и на пользователей системы.

Разграничение функциональных обязанностей подразумевает организацию работы с данными таким образом, что в каждой из ключевых стадий, составляющих единый критически важный с точки зрения целостности процесс, необходимо участие различных пользователей. Этим гарантируется, что один пользователь не может выполнить весь процесс целиком (или даже две его стадии) с тем, чтобы нарушить целостность данных. В обычной жизни примером воплощения данного принципа служит передача одной половины пароля для доступа к программе управления ядерным реактором первому системному администратору, а другой второму.

Как отмечено выше, принцип минимизации привилегий распространяется и на программы, и на пользователей. Последним, однако, на практике трудно назначить «теоретически достижимый» минимальный уровень привилегий по двум причинам. Во-первых, пользователи выполняют разнообразные задачи, требующие различных привилегий. Во-вторых, если строгое соблюдение принципа минимизации в отношении процессов связано с соображениями стоимости и производительности, то в отношении пользователей оно, скорее, затрагивает вопросы этики и морали, а также удобства и эффективности работы-это факторы, которые не поддаются точной количественной оценке. Поэтому пользователи будут, как правило, иметь несколько больше привилегий, чем им необходимо для выполнения конкретного действия в данный момент времени. А это открывает возможности для злоупотреблений. Аудит произошедших событий (включая возможность восстановления полной картины происшедшего) является превентивной мерой в отношении потенциальных нарушителей.

Принцип объективного контроля, согласно, также является одним из краеугольных камней политики контроля целостности. Суть данного принципа заключается в том, что контроль целостности данных имеет смысл лишь тогда, когда эти данные отражают реальное положение вещей.

Очевидно, что нет смысла заботиться о целостности данных, связанных с размещением боевого арсенала, который уже отправлен на переплавку. В связи с этим Кларк и Вилсон указывают на необходимость регулярных проверок, целью которых является выявление возможных несоответствий между защищаемыми данными и объективной реальностью, которую они отражают. Управление передачей привилегий необходимо для эффективной работы всей политики безопасности. Если схема назначения привилегий неадекватно отражает организационную структуру предприятия или не позволяет администраторам безопасности гибко манипулировать ею для обеспечения эффективности производственной деятельности, защита становится тяжким бременем и провоцирует попытки обойти ее там, где она мешает «нормальной» работе.

С некоторыми оговорками иногда в зарубежной научной литературе в основу контроля целостности закладывается и принцип обеспечения непрерывной работы (включая защиту от сбоев, стихийных бедствий и других форс-мажорных обстоятельств), который в классической теории компьютерной безопасности относится, скорее, к проблеме доступности данных.

В основу последнего девятого принципа контроля целостности - простота использования защитных механизмов—заложен ряд идей, призванных обеспечить эффективное применение имеющихся механизмов обеспечения безопасности. На практике зачастую оказывается, что предусмотренные в системе механизмы безопасности некорректно используются или полностью игнорируются системными администраторами по следующим причинам:

• неправильно выбранные производителем конфигурационные параметры по умолчанию обеспечивают слабую защиту;

• плохо разработанные интерфейсы управления защитой усложняют использование даже простых средств безопасности;

• имеющиеся средства безопасности не обеспечивают адекватный уровень контроля за системой;

• реализация механизмов безопасности плохо соответствует сложившемуся у администраторов интуитивному их пониманию;

• отдельные средства защиты плохо интегрированы в общую схему безопасности;

• администраторы недостаточно осведомлены о важности применения конкретных механизмов защиты и их особенностях.

Простота использования защитных механизмов подразумевает, что самый безопасный путь эксплуатации системы будет также наиболее простым, и наоборот, самый простой - наиболее защищенным.

Модель контроля целостности данных. Модель Кларка-Вилсона появилась в результате проведенного анализа реально применяемых методов обеспечения целостности документооборота в коммерческих организациях. Она изначально ориентирована на нужды коммерческих заказчиков, и более адекватна их требованиям, чем предложенная ранее коммерческая интерпретация модели целостности на основе решеток. Основные понятия рассматриваемой модели - это корректность транзакций и разграничение функциональных обязанностей. Модель задает правила функционирования компьютерной системы и определяет две категории объектов данных и два класса операций над ними. Все содержащиеся в системе данные подразделяются на контролируемые (КЭД) и неконтролируемые (НЭД) элементы соответственно[6]. Целостность первых обеспечивается моделью Кларка-Вилсона. Последние содержат информацию, целостность которой в рамках данной модели не контролируется (этим и объясняется выбор терминологии).

Далее, модель вводит два класса операций над элементами данных: процедуры контроля целостности (ПКЦ) и процедуры преобразования (ПП). Первые из них обеспечивают проверку целостности контролируемых элементов данных (КЭД), вторые изменяют состав множества всех КЭД (например, преобразуя элементы НЭД в КЭД). Наконец, модель содержит

девять правил, определяющих взаимоотношения элементов данных и процедур в процессе функционирования системы.

Правило С1. Множество всех процедур контроля целостности (ПКЦ) должно содержать процедуры контроля целостности любого элемента данных из множества всех КЭД.

Правило С2. Все процедуры преобразования (ПП) должны быть реализованы корректно в том смысле, что не должны нарушать целостность обрабатываемых ими КЭД. Кроме того, с каждой процедурой преобразования должен быть связан список элементов КЭД, которые допустимо обрабатывать данной процедурой. Такая связь устанавливается администратором безопасности.

Правило Е1. Система должна контролировать допустимость применения ПП к элементам КЭД в соответствии со списками, указанными в правиле С2.

Правило Е2. Система должна поддерживать список разрешенных конкретным пользователям процедур преобразования с указанием допустимого для каждой ПП и данного пользователя набора обрабатываемых элементов КЭД.

Правило С3. Список, определенный правилом С 2, должен отвечать требованию разграничения функциональных обязанностей.

Правило Е3. Система должна аутентифицировать всех пользователей, пытающихся выполнить какую-либо процедуру преобразования.

Правило С4. Каждая ПП должна записывать в журнал регистрации информацию, достаточную для восстановления полной картины каждого применения этой ПП. Журнал регистрации - это специальный элемент КЭД, предназначенный только для добавления в него информации.

Правило С5. Любая ПП, которая обрабатывает элемент НЭД, должна выполнять только корректные преобразования этого элемента, в результате которых НЭД превращается в КЭД.

Правило Е4. Только специально уполномоченное лицо может изменять списки, определенные в правилах С 2 и Е 2. Это лицо не имеет права выполнять какие-либо действия, если оно уполномочено изменять регламентирующие эти действия списки.

1.4. Особенности обеспечения целостности информации в сетях передачи данных

В компьютерной системе и, в частности в любой ОС, память разделена (по меньшей мере логически) на области, которые используют ее компоненты, а также программы пользователей. При этом необходимо обеспечить защиту областей памяти от вмешательства в них посторонних компонентов, т.е. разграничить доступ приложений к областям памяти, а в многозадачной среде и к областям памяти друг друга. Кроме того, необходимо решить проблему организации совместного доступа различных приложений к некоторым областям памяти. Обычно для этого используется один из трех подходов:

1) совместный доступ полностью исключен, возможно только монопольное использование области памяти;

2) допустимы только строго оговоренные типы доступа к содержимому данной области памяти, например, согласно таблице 1.2.

Таблица 1.2

	Чтение	Запись	Исполнение
Приложение 1	Да	Да	Да
Приложение 2	Да	Да	Нет
Приложение 3	Да	Нет	Нет

3) совместный доступ разрешен и ничем не ограничен;

4) совместный доступ может быть организован или к оригиналу области памяти, или предоставлением каждой программе индивидуальной

копии области. В последнем случае потребуется осуществлять синхронизацию обновлений области различными приложениями, а в первом - исключение одновременного изменения ее несколькими программами. Следует также учесть, что в совместное пользование могут быть предоставлены не только данные, но и исполняемый код. Таким образом, в задачи АС по предоставлению областей памяти в совместное пользование входят:

- организация последовательного, взаимоисключающего доступа нескольких программ к совместно используемым объектам;
- ограничение возможностей совместно используемых программ по манипулированию информацией различной ценности. Кратко рассмотрим основные способы защиты памяти.

Особенности обеспечения целостности информации барьерных адресов. Барьерный адрес указывает на начало пользовательской области памяти, отделяя ее от области памяти, в которой размещается программы АС и ее данные (как правило, это области младших адресов). Это напоминает забор, построенный дачником, чтобы отгородиться от назойливого соседа.

В предположении, что АС размещена в области младших адресов, можно определить функционирование механизма барьерного адреса следующим образом. При каждом обращении пользовательской программы к памяти адрес запрашиваемой ячейки сравнивается с барьерным. Допустимыми считаются обращения к ячейкам памяти с адресами, большими барьерного адреса. Программа, которая пытается обратиться к памяти АС (т.е. к ячейке с адресом, меньшим барьерного) аварийно завершается выдачей пользователю сообщения об ошибке. Значение барьерного адреса может быть представлено константой, записанной в поддерживающей АС аппаратуре, что накладывает ограничения на максимальный размер самой АС или может привести к неэффективному использованию ресурсов памяти, если размер АС достаточно мал и она оставляет неиспользованной часть отведенного ей адресного пространства.

Другой, более гибкий способ задания барьерного адреса его хранение в специальном регистре, значение которого устанавливается привилегированной командой в начале работы АС и может динамически изменяться в соответствии с ее потребностями. Последний подход накладывает ограничение на механизмы адресации, исполняемыми в данной АС программами. Физические адреса данных загруженной в память программы определяются сложением логических адресов со значением барьерного адреса. Т.е. предполагается, что логическое адресное пространство программы начинается с нулевого адреса, соответствующего ячейке, начиная с которой программа размещается в памяти. Если определение физических адресов на основе заданных в программе логических адресов происходит на этапе компиляции, то для корректной работы программы необходимо, чтобы известный на этапе компиляции барьерный адрес оставался неизменным на протяжении всей работы программы и при каждом повторном ее запуске. Другими словами, использование программ возможно только, если барьерный адрес задан в качестве константы (в АС или поддерживающей ее аппаратной платформе), и программа загружается в фрагмент памяти, адрес которого строго задан. В противном случае, изменение барьерного адреса потребовало бы перекомпиляции всех написанных для данной АС программ.

В том случае, когда определение физических адресов происходит в момент загрузки программы в память, подобного строгого ограничения нет. Однако сохраняется требование неизменности барьерного адреса в процессе выполнения программы. При изменении барьерного адреса программу нужно загрузить в память повторно. В обоих рассмотренных случаях на этапе выполнения программы адреса ее данных задавались в абсолютном формате (загруженная программа содержала реальные значения физических адресов), что накладывало ограничения на способ изменения барьерного адреса АС. Более гибкий способ перераспределения памяти между АС и приложениями обеспечивает механизм динамических областей памяти.

Особенности обеспечения целостности информации в динамических областях памяти. Физические адреса программных данных могут вычисляться не только, как описывалось выше, в момент компиляции или загрузки программы, но и непосредственно в процессе ее выполнения.

При этом логические адреса данных, как и раньше, начинаются с нулевого, а при обращении к ячейке памяти ее физический адрес вычисляется сложением логического со значением барьерного адреса. Таким образом, обеспечивается возможность изменения последнего в ходе выполнения программы, а программы АС и приложение занимают динамические области памяти.

Рассмотренные выше способы защиты реализуют только разграничение доступа к памяти АС и приложений. Однако в многозадачной системе необходимо также отдельно защитить данные каждого приложения. Такие механизмы рассматриваются ниже.

Особенности обеспечения целостности информации в адресных регистрах. Доступная пользовательской программе область памяти может быть ограничена парой хранящихся в регистрах значений: начальным и конечным адресом области. При этом каждой программе отводится отдельная область памяти и отдельная пара адресных регистров. Один из вариантов использования адресных регистров -это хранение в них начального и конечного физических адресов области памяти приложения. В этом случае при каждом обращении программу к памяти проверяется принадлежность адреса заданному содержимым этих регистров промежутку.

Другой способ применения адресных регистров - задание в них базового и предельного адресов области памяти приложения, причем первый является физическим, а второй - логическим. Адрес, по которому происходит обращение к памяти, сначала сравнивается с содержимым регистра, задающего предельно допустимый логический адрес (напоминание: адресация в логическом пространстве начинается с нуля).

Если указанный в программе адрес меньше предельного, для получения физического адреса к нему прибавляется значение базового. В противном случае программа аварийно завершается. Такой способ в отличие от предыдущего допускает динамическое перемещение программы в памяти. Более надежные способы защиты памяти на основе адресных регистров предполагают использование двух пар регистров для каждой про граммы: отдельно для фрагмента кода и данных. При этом запись в первый фрагмент может быть запрещена для защиты кода от изменения (как предполагается, непреднамеренного). Все уже рассмотренные способы защиты памяти обладают одной общей особенностью, а именно, обеспечивают защиту выделенной области памяти, состоящей из последовательно расположенных ячеек. Однако нередко возникает необходимость более тонко разграничивать доступ к памяти, что можно реализовать с помощью ключей доступа. Ключ доступа - это устанавливаемый операционной системой атрибут отдельной ячейки памяти, на основе которого затем осуществляется проверка допустимости каждого обращения. Другая особенность механизма защиты с применением адресных регистров связана с организацией совместного использования областей памяти. Две пары регистров (для кода и данных, как говорилось выше) позволяют эффективно организовать совместное использование только фрагментов кода (запись в которые запрещена). Для защиты фрагмента данных от непреднамеренного искажения в результате операций записи необходимы дополнительные средства (адресные регистры позволяют контролировать только сам факт доступа к области памяти, но не конкретный способ осуществления доступа: чтение или запись). Более совершенные средства защиты памяти обеспечиваются механизмами страничной организации памяти и сегментации[7].

Страницы и сегменты памяти. При страничной организации памяти все адресное пространство разделяется на блоки фиксированного размера (страницы). Проверка допустимости адресов и преобразование логических адресов в физические при обращениях к памяти осуществляется с помощью

таблицы страниц, каждая запись которой содержит начальный адрес расположения страницы в памяти. Логический адрес имеет формат <номер страницы, смещение>. По номеру страницы определяется физический адрес ее первой ячейки, к которому затем прибавляется смещение. С каждой страницей ассоциирован ключ доступа, определяющий набор допустимых операций (чтение, запись, исполнение). Такая схема организации памяти поддерживает совместное использование страниц: в таблицах страниц отдельных приложений могут быть описаны одни и те же страницы физической памяти. Однако более изощренное управление доступом в страничной модели затруднено из-за того, что на одной странице могут находиться различные по своей природе объекты (например, код и данные некоторой программы). Смысл сегментной организации памяти заключается в том, чтобы предоставить индивидуальные (возможно, неодинаковые по размеру) области памяти логически различным частям программы. Например, в отдельные сегменты могут быть помещены данные с различным типом доступа или код основной части программы и вызываемых ею подпрограмм. Преобразование логических адресов в физические происходит на основе содержимого таблицы сегментов, куда заносятся базовый и предельный адреса каждого сегмента, (они обсуждались при описании функционирования адресных регистров). Логический адрес имеет формат <номер сегмента, смещение>, похожий на формат, применяемый в страничной адресации. Однако теперь каждый сегмент содержит однородные объекты, которым необходим одинаковый уровень защиты, так же как и выше, обеспечиваемый ключом доступа. Возможно совместное использование сегментов данных и кода, отдельные участки которого (например, процедуры и функции) могут быть предоставлены в совместное использование независимо от основной программы.

2. Основная часть. Обеспечение целостности информации с использованием избыточных кодов и роль хэш-функции в обеспечениях их безопасности

2.1. Классификации избыточных кодов и их параметры

Циклические коды являются подклассом в классе линейных кодов, удовлетворяющим дополнительному сильному структурному требованию. В силу этой структуры поиск хороших кодов, контролирующих ошибки, в классе циклических кодов оказался наиболее успешным. При этом в качестве математического аппарата, облегчающего поиск хороших кодов, была использована теория полей Галуа[8]. Вне класса циклических кодов теория полей Галуа помогает мало; большинство завершённых построений, использующих идеи этой теории, относятся к циклическим кодам.

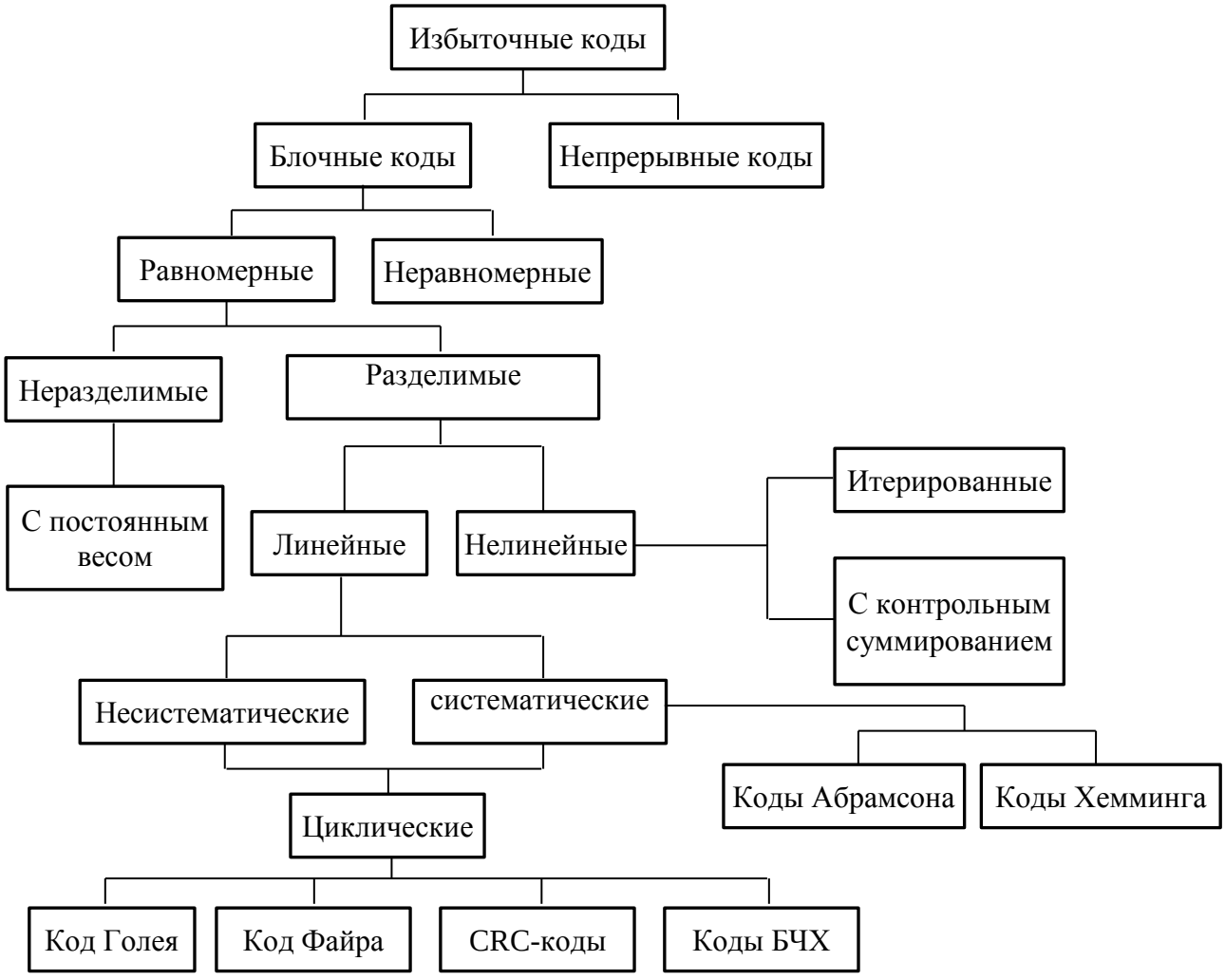


Рис.2.1. Классы циклических кодов

Важность циклических кодов обусловлена также, тем, что заложенные в основу их определения идеи теории полей Галуа приводят к процедурам кодирования и декодирования, эффективным как с алгоритмической, так и с вычислительной точки зрения. Поэтому нужно знать свойства полей Галуа.

Полем называется множество с двумя определёнными на нём операциями: сложением и умножением.

Существуют следующие примеры полей:

1. R: множество вещественных чисел;

2. C: множество комплексных чисел;

3. Q: множество рациональных чисел.

Все эти поля содержат бесконечное число элементов. Но нас интересуют поля с конечным числом элементов. Конечное поле, названное в честь французского математика Галуа обозначается GF(q), где q-это конечное множество элементов обладающих свойством поля.

Циклические коды обычно применяются в системах с последовательной передачей двоичных разрядов контролируемого кода, в частности при записи и считывании информации в накопителях на магнитных дисках. При незначительном увеличении избыточного оборудования такие коды способны с высокой эффективностью обнаруживать не только одиночные, но и групповые сигналы.

Особенность циклического кода заключается в том, что если n-значная информационная комбинация $a_0a_1a_2\ldots a_n$ принадлежит к данному коду, то и комбинация $a_n,a_0,a_1\ldots a_{n-1}$ в которой произведена циклическая перестановка символов, также принадлежит этому коду.

Принцип построения циклического кода состоит в следующем. Любая двоичная комбинация может быть представлена в виде полинома (n-1)-й степени некоторой переменной x, коэффициентами которого являются двоичные символы соответствующих разрядов числа. Например, двоичное число 11001 представляет i полином четвертой степени: $1\cdot x^4+1\cdot x^3+0\cdot x^2+0\cdot x^1+1\cdot x^0$.

Над такими полиномами можно произвести все математические действия в соответствии с известными законами алгебры, за исключением сложения, которое в отличие от обычной алгебры осуществляется по модулю 2 (поразрядное сложение без учета переносов).

Циклический код n-значного числа строится путем добавления к m информационным разрядам к контрольных разрядов (n=m+k). Кодирование заключается в составлении на основе m-разрядной информационной комбинации G(x) такого многочлена F(x), который должен делиться без остатка на заранее выбранный порождающий полином P(x) степени к, с помощью которого и происходит образование циклического кода.

Формирование кодовой комбинации F(x) осуществляется следующим образом. Информационный многочлен G(x), подлежащий кодированию, вначале умножается на x^k, что соответствует сдвигу кодируемого числа на k разрядов влево (в сторону старших разрядов). В результате деления многочлена $G(x) \cdot x^k$ на порождающий полином P(x) образуется частное Q(x) и остаток R(x), т.е. $G(x) \cdot x^k = Q(x)P(x)$ ØR(x). Если искомый кодовый многочлен образовать путем присоединения остатка R(x) к младшим разрядам сдвинутого, информационного кода $G(x)x^k$, то такой многочлен должен без остатка делиться на пораждающий полином P(x), т.е.

$F(x) = G(x)x^k$ ØR(x) = Q(x)P(x).

Таким образом, используя остаток R(x) в качестве контрольных разрядов в многочлене F(x) и производя после каждой передачи деление принятой кодовой комбинации на порождающий полином, можно по отсутствию остатка судить о возможных искажениях передаваемого кода.

Выбор числа членов и степени старшего разряда порождающего полинома P(x) требует отдельного рассмотрения. Здесь приведено лишь вид этого полинома для различной разрядности кодовых комбинаций:

Таблица 2.1

Порождающий полином P(x)	Число разрядов		
	$n \leq$	m	k

x^3+x+1	7	4	3
x^4+x+1	15	11	4
x^5+x^2+1	31	26	5

Циклическое кодирование с помощью данных полиномов P(x) позволяет обнаруживать и исправлять одиночные ошибки, а также обнаруживать двойные ошибки. Рассмотрено этот процесс на конкретном примере.

Пусть дано двоичное число 11010, что соответствует информационному многочлену четвертой степени $G(x) = x^4 + x^3 + x$. Для данного случая используется пораждающий полином третьей степени $P(x) = x^3 + x + 1$, соответствующий коду 1011. Сдвинутый на 3 разряда влево (к = 3) информационный код делим на код 1011 порождающего полинома. При этом деление производится путем последовательного вычитания по модулю 2 (или сложения, что то же самое) делителя из делимого и получающихся разностей до тех пор, пока разность не будет иметь более низкую степень, чем делитель. Эта последняя разность и есть искомый остаток, который нужно дописать в младшие разряды информационной комбинации.

Рассмотренные способ кодирования, часто объединяемые под общим названием помехоустойчивого кодирования, всегда основаны на использовании информационной избыточности. Рабочая информация в КС дополняется определенным объемом специальной контрольной информации. Наличие этой контрольной информации позволяет обнаруживать ошибки и даже исправлять их. Чем больше используется контрольной информации, тем шире возможности кода по обнаружению и исправлению ошибок. Однако с увеличением избыточности всегда растут дополнительные затраты на использование необходимой контрольной аппаратуры, поэтому в каждом конкретном случае необходимо учитывать и соразмерять эти характеристики КС.

Поскольку ошибки в работе КС часто являются следствием отказов технических средств, то, используя помехоустойчивое кодирование, можно парировать часть отказов и создавать так называемые отказоустойчивые системы. Наиболее совершенными, устойчивыми к отказам, являются адаптивные системы. В них достигается разумный компромисс между уровнем избыточности, вводимым для обеспечения устойчивости системы к отказам, и эффективностью использования таких систем по назначению. В адаптивных системах реализуется так называемый принцип элегантной деградации. Этот принцип предполагает сохранение работоспособности системы в случаях отказов ее элементов при некотором снижении эффективности системы в целом. Адаптивные системы содержат программно-аппаратные средства для автоматического контроля работоспособности элементов системы и осуществляющие ее реконфигурацию при возникновении отказов элементов. При реконфигурации восстанавливается необходимая информация (при ее утрате), отключается отказавший элемент, осуществляется изменение связей и режимов работы системы.

2.2. CRC коды, алгоритмы и их использование

Алгоритм CRC базируется на свойствах деления с остатком двоичных многочленов, то есть многочленов над конечным полем GF(2). Значение CRC является по сути остатком от деления многочлена, соответствующего входным данным, на некий фиксированный порождающий многочлен (полином).

Каждой конечной последовательности битов$a_0, a_1, \dots a_{N-1}$ взаимно однозначно сопоставляется двоичный многочлен $\sum_{n=0}^{N-1} a_n x^n$, последовательность коэффициентов которого представляет собой исходную последовательность. Например, последовательность битов 0,1,0,1,1,0,1 соответствует многочлену

$$P(x) = 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0$$
$$= x^6 + x^4 + x^3 + x^1$$

Нетрудно видеть, что количество различных многочленов степени меньшей N равно 2^N, что совпадает с числом всех двоичных последовательностей длины N.

Значение CRC с порождающим многочленом $G(x)$ степени N определяется как битовая последовательность длины N, представляющая многочлен $R(x)$, получившийся в остатке при делении многочлена $P(x)$, представляющего входной поток бит, на многочлен $G(x)$:

$R(x) = P(x) \cdot x^N mod G(x)$, где

R(x) — многочлен, представляющий значение CRC.

P(x) — многочлен, коэффициенты которого представляют входные данные.

G(x) — порождающий многочлен.

$N = \deg G(x)$ — степень порождающего многочлена.

Умножение x^N осуществляется приписыванием N нулевых битов к входной последовательности, что улучшает качество хеширования для коротких входных последовательностей.

При *делении с остатком* степень многочлена-остатка строго меньше степени многочлена-делителя, то есть при делении на многочлен $G(x)$ степени N можно получить 2^N различных остатков от деления. При «правильном» выборе порождающего многочлена $G(x)$, остатки от деления на него будут обладать нужными свойствами хеширования - хорошей перемешиваемостью и быстрым алгоритмом вычисления. Второе обеспечивается тем, что степень порождающего многочлена обычно пропорциональна длине байта или машинного слова (например 8, 16 или 32).

Операция деления на примитивный полином также эквивалентна следующей схеме:

Пусть выбран примитивный полином, задающий цикл де Брейна 0010111001011100… и блок данных 0111110, построена таблица, верхняя

строка заполнена блоком данных, а нижние строки — смещения на 0,1,2 бит цикла де Брейна

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Тогда контрольная сумма будет равна операции XOR тех столбцов, над которыми в верхней строке расположена 1. В этом случае, 010 xor 101 xor 011 xor 111 xor 110 = 101 (CRC).

Ниже представлены реализации получения некоторых CRC для многочленов степени 8 (CRC-8), 16 (CRC-16) и 32 (CRC-32).

Классификация реализаций алгоритмов CRC. Для точного указания, как именно рассчитывается CRC, чаще всего приходится полностью приводить алгоритм её расчёта[9]. На самом деле достаточно указать ряд параметров, точно описывающих конкретный частный алгоритм CRC (если это на самом деле CRC).

В модели алгоритма CRC Rocksoft, получившей некоторое хождение, используются следующие параметры:

Name: Это имя, присвоенное данному алгоритму.

Width: Степень алгоритма, выраженная в битах. Она всегда на единицу меньше длины полинома, но равна его степени.

Poly: Собственно полином. Это битовая величина, которая для удобства может быть представлена шестнадцатеричным числом. Старший бит при этом опускается (он всегда 1). Например, если используется полином 10110, то он обозначается числом «06h». Важной особенностью данного параметра является то, что он всегда представляет собой необращенный полином, младшая часть этого параметра во время вычислений всегда является наименее значащими битами делителя.

Init: Этот параметр определяет исходное содержимое регистра на момент запуска вычислений. Данный параметр указывается шестнадцатеричным числом.

RefIn(Revert): Логический параметр. Если он имеет значение False, байты сообщения обрабатываются, начиная с 7 бита, который считается наиболее значащим, а наименее значащим считается бит 0 (сдвиг налево). Если параметр имеет значение True («Истина»), то каждый байт перед обработкой обращается (сдвиг направо).

RefOut: Логический параметр. Если он имеет значение False, то конечное содержимое регистра сразу передается на стадию XorOut, в противном случае, когда параметр имеет значение True, содержимое регистра обращается перед передачей на следующую стадию вычислений.

XorOut: W битное значение, обозначаемое шестнадцатеричным числом. Оно комбинируется с конечным содержимым регистра (после стадии RefOut), прежде чем будет получено окончательное значение контрольной суммы.

Check: Это поле, собственно, не является частью определения алгоритма, данное поле служит контрольным значением, которое может быть использовано для слабой проверки правильности реализации алгоритма. Поле содержит контрольную сумму, рассчитанную для ASCII строки «123456789» (шестнадцатеричные значение «313233…»).

После определения всех этих параметров, можно точно описать особенности применённого CRC алгоритма.

Таблица 2.2

Примеры спецификаций некоторых алгоритмов CRC

Name : CRC 16 Width : 16 Poly : 8005 Init : 0000 RefIn : True RefOut : True XorOut : 0000	Name : CRC 16/CITT Width : 16 Poly : 1021 Init : FFFF RefIn : False RefOut : False XorOut : 0000	Name : XMODEM Width : 16 Poly : 8408 Init : 0000 RefIn : True RefOut : True XorOut : 0000

Check : BB3D		
Name : ARC Width : 16 Poly : 8005 Init : 0000 RefIn : True RefOut : True XorOut : 0000	Name : CRC 32 Width : 32 Poly : 04C11DB7 Init : FFFFFFFF RefIn : True RefOut : True XorOut : FFFFFFFF Check : CBF43926	

Наиболее используемые и стандартизованные CRC. В то время, как циклические избыточные коды являются частью стандартов, сами они не стандартизированы в плане адаптации одного алгоритма для конкретной степени полинома: существуют три описания полинома для CRC-12, десять противоречивых определений CRC-16 и четыре - CRC-32. Наиболее широко используемые полиномы не являются наиболее эффективными из всех возможных. Между 1993 и 2004, Koopman, Castagnoli и другие исследовали пространство полиномов до 16 бит включительно, и 24 и 32 бит, найдя примеры, которые дают гораздо большую производительность (в смысле Hamming distance для данных заданного размера), чем полиномы предшествовавших протоколов, и опубликовав лучшие из них с целью улучшения способности к выявлению ошибок будущих стандартов. В настоящее время в протоколе ISCSI используется один из результатов этого исследования.

Самый популярный, рекомендуемый IEEE полином для CRC-32, используемый в Ethernet, FDDI и др., является генератором кода Хемминга и был выбран, основываясь на его производительности и способности выявления ошибок передачи данных.

Таблица 2.3

Тип	Полиноминал	Представление нормальный/полностью измененный/перемена взаимных

CRC-1	$x + 1$ (в аппаратуре)	0x1 / 0x1 / 0x1
CRC-4-ITU	$x^4 + x + 1$ (ITU G.704)	0x3 / 0xC / 0x9
CRC-5-EPC	$x^5 + x^3 + 1$ (Gen 2 RFID)	0x09 / 0x12 / 0x14
CRC-5-ITU	$x^5 + x^4 + x^2 + 1$ (ITU G.704)	0x15 / 0x15 / 0x1A
CRC-5-USB	$x^5 + x^2 + 1$ (в пакете USB токен)	0x05 / 0x14 / 0x12
CRC-6-ITU	$x^6 + x + 1$ (ITU G.704)	0x03 / 0x30 / 0x21
CRC-7	$x^7 + x^3 + 1$ (телекоммуникационные системы, ITU-T G.707, ITU-T G.832, SD)	0x09 / 0x48 / 0x44
CRC-8-ATM	$x^8 + x^2 + x + 1$ (ATM HEC)	0x07 / 0xE0 / 0x83
CRC-8-CCITT	$x^8 + x^7 + x^3 + x^2 + 1$ (1-в проводе шину)	0x8D / 0xB1 / 0xC6
CRC-8-Dallas/Maxim	$x^8 + x^5 + x^4 + 1$ (1-в проводе шину)	0x31 / 0x8C / 0x98
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	0xD5 / 0xAB / 0xEA[1]
CRC-8-SAE	$x^8 + x^4 + x^3 + x^2 + 1$	0x1D / 0xB8 / 0x8E
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	0x233 / 0x331 / 0x319
CRC-11	$x^{11} + x^9 + x^8 + x^7 + x^2 + 1$ (FlexRay[12])	0x385 / 0x50E / 0x5C2
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$ (в телекоммуникационных системы)	0x80F / 0xF01 / 0xC07

CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$	0x4599 / 0x4CD1 / 0x62CC
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ (Bisync, Modbus, USB, ANSI X3.28)	0x8005 / 0xA001 / 0xC002
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ (X.25, HDLC, XMODEM, Bluetooth, SD*T*)	0x1021 / 0x8408 / 0x8810[1]
CRC-16-T10-DIF	$x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (SCSI DIF)	0x8BB7[16] / 0xEDD1 / 0xC5DB
CRC-16-DNP	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$ (DNP, IEC 870, M-Bus)	0x3D65 / 0xA6BC / 0x9EB2
CRC-16-Fletcher	Контрольная сумма Флетчера	Используется Adler-32 A & B CRC
CRC-24	$x^{24} + x^{22} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^3 + x + 1$ (FlexRay[12])	0x5D6DCB / 0xD3B6BA / 0xAEB6E5
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$ (OpenPGP)	0x864CFB / 0xDF3261 / 0xC3267D
CRC-30	$x^{30} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$ (CDMA)	0x2030B9C7 / 0x38E74301 / 0x30185CE3
CRC-32-Adler	Adler-32	Adler-32
CRC-32-IEEE 802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (V.42, MPEG-2, PNG, контрольная сумма POSIX)	0x04C11DB7 / 0xEDB88320 / 0x82608EDB

CRC-32C (Castagnoli)	$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$ (iSCSI, в платёжная система G.hn)	0x1EDC6F41 / 0x82F63B78 / 0x8F6E37A0
CRC-32K (Koopman)	$x^{32} + x^{30} + x^{29} + x^{28} + x^{26} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{11} + x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1$	0x741B8CD7 / 0xEB31D82E / 0xBA0DC66B
CRC-32Q	$x^{32} + x^{31} + x^{24} + x^{22} + x^{16} + x^{14} + x^8 + x^7 + x^5 + x^3 + x + 1$ (в авиации ; AIXM)	0x814141AB / 0xD5828281 / 0xC0A0A0D5
CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$ (HDLC — ISO 3309)	0x000000000000001B / 0xD800000000000000 / 0x8000000000000000D
CRC-64-ECMA-182	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$	0x42F0E1EBA9EA3693 / 0xC96C5795D7870F42 / 0xA17870F5D4F51B49

Циклический избыточный код CRC-4. Из всей совокупности методов контроля с использованием циклической структуры группового сигнала наибольшее распространение получил метод контроля первичных цифровых трактов CRC-4.

При передаче потока Е1 по сети связи, включающей в себя ряд систем передачи плезиохронной и синхронной цифровых иерархий, прозрачных для его прохождения, возникает необходимость проверки качественных показателей первичного цифрового канала на всём его протяжении.

Из 265 бит, образующих стандартный цикл 2-мегабитного сигнала, сигналы с 1-ого по 31-й КИ (канальный интервал) являются случайными. Поэтому из всего группового сигнала можно идентифицировать только 7 бит циклового синхросигнала, что позволяет обнаруживать битовые ошибки без остановки связи, однако указанные 7 бит составляют только 1,4 % от общего

объёма передаваемой информации. Проблема обеспечения контроля ошибок в остальных 31 каналах оптимально решается путём использования метода контроля с использованием избыточности циклического сигнала CRC-4.

CRC-4 был определён в 1980-х годах рекомендацией МСЭ-Т, но широкое распространение получил только в последнее время вследствие трудностей схемотехнической реализации, которую можно осуществить только методами микросхемотехники. Сущность метода состоит в том, что цифровой сигнал разбивается на группы, получившие название блоков или субсверхциклов. На передающем конце тракта производится подсчёт суммы символов блока, эта информация в составе группового сигнала передаётся на приёмный конец тракта, где подсчёт суммы символов повторяется, и результаты подсчёта на передающем и приёмном концах сравниваются. Совпадение результатов интерпретируется как отсутствие ошибок при передаче блока. Расхождение результатов говорит о наличии ошибок при передаче данного блока. Для передачи результатов сравнения в обратном направлении, на передающий конец тракта, используются свободные биты служебных канальных интервалов группового сигнала обратного направления.

16 следующих друг за другом циклов образуют сверхцикл, который, в свою очередь, делится на два субсверхцикла (1-й и 2-й) по 8 циклов каждый. Таким образом, временной интервал CRC-4 равен 16 × 125 мкс = 2 мс. Для формирования сигнала CRC-4 сумма бинарных символов каждого субсверхцикла делится на полином четвертой степени ($x^4 + x + 1$). Результат деления, представляющий собой 4 бинарных символа, вводится в групповой сигнал в позициях от С1 до С4. Приёмная сторона использует аналогичный метод для того, чтобы затем сравнить кодовое слово, поступающее от передатчика, с результатом, полученным на приёмном конце. Если указанные слова различны, значит субсверхцикл, равный 2048 битам, был передан с ошибками. Преимуществом этого метода является то, что с его помощью можно контролировать цифровой поток без остановки связи и

независимо от его содержания. Вместе с тем, при использовании указанного метода невозможно точно указать, какие биты из тысяч переданных были приняты с ошибками. Сверх цикловые сигнал CRC-4 служит для того, чтобы можно было обеспечить синхронизацию по битам от С1 до С4. Биты Е (Е1 и Е2 для 1-го и 2-го субсверх циклов) инвертируются для сохранения структуры сверхцикла в моменты обнаружения ошибок. Таким образом, приёмная сторона может информировать передающую сторону об обнаружении ошибок передачи. После того, как установится цикловая синхронизация, сигналы CRC-4 будут передаваться непрерывно. Потеря синхронизации CRC-4 происходит тогда, когда более чем 914 сигналов CRC-4, передаваемые в течение 1 секунды, не будут соответствовать нормированным.

Аналогично организуется и проверка цифровых трактов других ступеней иерархии. Меняется только величина блоков и степень полинома: 6-я степень для CRC-6, используемого для контроля ИКМ-120, 8-я— для CRC-8, используемого для контроля ИКМ-480.

Формализованный алгоритм расчёта CRC16. Для получения контрольной суммы, необходимо сгенерировать полином. Основное требование к полиному: его степень должна быть равна длине контрольной суммы в битах. При этом старший бит полинома обязательно должен быть равен «1».Из файла берется первое слово[10]. Если старший бит в слове «1», то слово сдвигается влево на один разряд с последующим выполнением операции XOR. Соответственно если старший бит в слове «0», то после сдвига операция XOR не выполняется. После сдвига (умножения) теряется старый старший бит, а младший бит освобождается (обнуляется). На место младшего бита загружается очередной бит из файла. Операция повторяется до тех пор, пока не загрузится последний бит файла.

2.3. Функциональная роль хэш-функции в обеспечения целостности информации в сетях передачи данных

Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция h(M) принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение h(M)-H фиксированной длины. Обычно хэшированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Следует отметить, что значение хэш-функции h(M) сложным образом зависит от документа M и не позволяет восстановить сам документ M.

Хэш-функция должна удовлетворять целому ряду условий:

- хэш-функция должна быть чувствительна к всевозможным изменениям в тексте M, таким как вставки, выбросы, перестановки и т.п.;
- хэш-функция должна обладать свойством необратимости, то есть задача подбора документа M', который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;
- вероятность того, что значения хэш-функции двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала.

Большинство хэш-функций строится на основе однонаправленной функции f(H), которая образует выходное значение длиной n при задании двух входных значений длиной n. Этими входами являются блок исходного текста M и хэш-значение $H_i(M_i)$ предыдущего блока текста (рис.2.2):

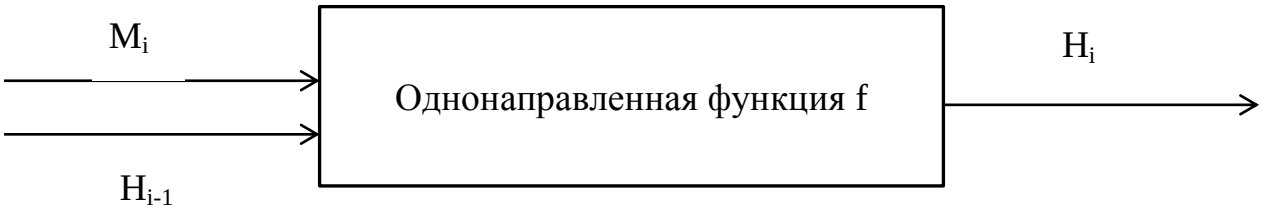


Рис.2.2. Построение однонаправленной хэш-функции

$$H_i = f(M_i, H_{i-1}).$$

Хэш-значение, вычисляемое при вводе последнего блока текста, становится хэш-значением всего сообщения M.

В результате однонаправленная хэш-функция всегда формирует выход фиксированной длины n (независимо от длины входного текста).

Таблица 2.4

MD 5

Название	MD 5
Создан	1991 г
Опубликован	Апрел 1992 г
Размер хэша	128 бит
Число раундов	4
Тип	Хеш-функция

MD5 (англ. *Message Digest 5*) - 128-битный алгоритм хэширования, предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины. Является улучшенной в плане безопасности версией MD4. Зная MD5-образ (называемый также MD5-хеш или MD5-дайджест), невозможно восстановить входное сообщение, так как одному MD5-образу могут соответствовать разные сообщения[11]. Используется для проверки подлинности опубликованных сообщений путём сравнения дайджеста сообщения с опубликованным. Эту операцию называют «проверка хэша» (hashcheck). Из-за небольшого размера хеша в 128 бит, можно рассматривать birthday атаки. В марте 2004 года был запущен проект MD5CRK с целью обнаружения уязвимостей алгоритма, используя birthday атаки.

Алгоритм MD5. На вход алгоритма поступает входной поток данных, хэш которого необходимо найти. Длина сообщения может быть любой (в том числе нулевой). Запишем длину сообщения в L. Это число целое и неотрицательное. Кратность каким-либо числам необязательна. После поступления данных идёт процесс подготовки потока к вычислениям.

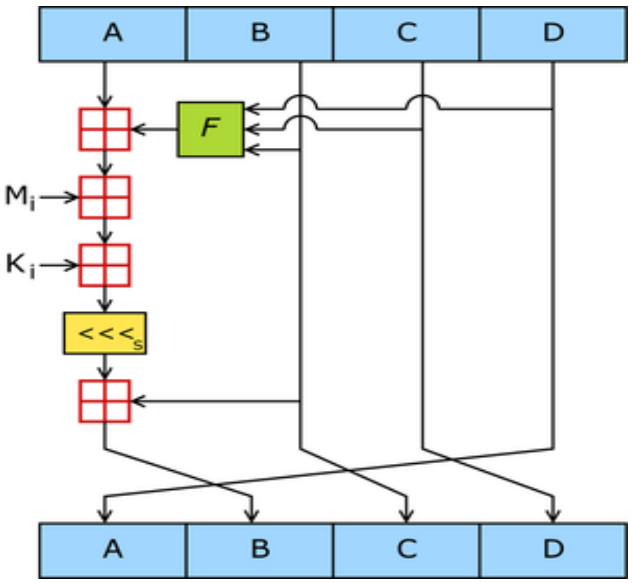


Рис.2.3. Схема работы алгоритма MD5

Ниже приведены 5 шагов алгоритма:

Шаг 1. Выравнивание потока. Входные данные выравниваются так, чтобы их размер был сравним с 448 по модулю 512 (L’ = 512×N + 448). Сначала дописывают единичный бит в конец потока, затем необходимое число нулевых бит (выравнивание происходит, даже если длина уже конгруэнтна - сравнима с 448).

Шаг 2. Добавление длины сообщения. В оставшиеся 64 бита дописывают 64-битное представление длины данных (количество бит в сообщении) до выравнивания. Если длина превосходит $2^{64} - 1$, то дописывают только младшие биты. После этого длина потока станет кратной 512. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит.

Шаг 3. Инициализация буфера. Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения шестнадцатеричными числами:

A = 01 23 45 67;

B = 89 AB CD EF;

C = FE DC BA 98;

D = 76 54 32 10.

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

Определено ещё функции и константы, которые нам понадобятся для вычислений.

Потребуются 4 функции для четырёх раундов. Введём функции от трёх параметров - слов, результатом также будет слово.

1 раунд $FunF(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$.

2 раунд $FunG(X,Y,Z) = (X \wedge Z) \vee (\neg Z \wedge Y)$.

3 раунд $FunH(X,Y,Z) = X \oplus Y \oplus Z$.

4 раунд $FunI(X,Y,Z) = Y \oplus (\neg Z \vee X)$.

Определено таблицу констант T[1..64] — 64-элементная таблица данных, построенная следующим образом: $T[i] = int(4294967296 * |sin(i)|)$, где $4294967296 = 2^{32}$.[3]

Выровненные данные разбиваются на блоки (слова) по 32 бита, и каждый блок проходит 4 раунда из 16 операторов. Все операторы однотипны и имеют вид: [abcd k s i], определяемый как

$a = b + ((a + Fun(b,c,d) + X[k] + T[i]) <<< s)$

где X - блок данных. X[k] = M [n*16 + k], где k - номер 32-битного слова из n-го 512-битного блока сообщения, и s - циклический сдвиг влево на s бит полученого 32-битного аргумента.

Шаг 4. Вычисление в цикле. Занесено в блок данных элемент n из массива. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый).

AA = A

BB = B

CC = C

DD = D

Раунд 1

/*[abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]

[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

Раунд 2

/*[abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]

[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]

[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]

[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

Раунд 3

/*[abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

Раунд 4

/*[abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

[ABCD 14 6 65][DABC 13 10 66][CDAB 4 15 67][BCDA 16 24 68]

Суммируем с результатом предыдущего цикла:

A = AA + A

B = BB + B

C = CC + C

D = DD + D

После окончания цикла необходимо проверить, есть ли ещё блоки для вычислений. Если да, то изменяем номер элемента массива (n++) и переходим в начало цикла.

Шаг 5. Результат вычислений. Результат вычислений находится в буфере ABCD, это и есть хеш. Если вывести слова в обратном порядке DCBA, то мы получим наш MD5 хеш.

MD5-хеши. Хеш содержит 128 бит (16 байт) и обычно представляется как последовательность из 32 шестнадцатеричных цифр.

Несколько примеров хеша:

MD5(«md5») = 1bc29b36f623ba82aaf6724fd3b16718

Даже небольшое изменение входного сообщения (в нашем случае на один бит: ASCII символ «5» с кодом $0x35_{16} = 000110101_2$ заменяется на символ «4» с кодом $0x34_{16} = 000110100_2$) приводит к полному изменению хеша. Такое свойство алгоритма называется лавинным эффектом.

MD5(«md4») = c93d3bf7a7c4afe94b64e30c2ce39f4f

Пример MD5-хеша для «нулевой» строки:

MD5(«») = d41d8cd98f00b204e9800998ecf8427e

Криптоанализ. На данный момент существуют несколько видов «взлома» хешей MD5 - подбора сообщения с заданным хешем:

1. Перебор по словарю.
2. Brute-force.
3. RainbowCrack.

Атаки переборного типа. Для полного перебора или перебора по словарю можно использовать программы PasswordsPro, MD5BFCPF, John the Ripper. Для перебора по словарю существуют готовые словари.

RainbowCrack— еще один метод взлома хеша. Он основан на генерировании большого количества хешей из набора символов, чтобы по получившейся базе вести поиск заданного хеша[12]. Хотя генерация хешей занимает много времени, зато последующий взлом производится очень быстро. Rainbow-таблицы можно найти как готовые, так и сгенерировать

самостоятельно.Важно отметить, что наличие у MD5 коллизий упрощает (но не усложняет) взлом многих приложений MD5, когда по заданной хеш-сумме достаточно найти любое значение входных данных ей соответствующее.

Примеры использования. MD5 позволяет получать относительно надёжный идентификатор для блока данных. Такое свойство алгоритма широко применяется в разных областях[13]. Оно позволяет искать дублирующиеся файлы на компьютере, сравнивая MD5 файлов, а не их содержимое. Как пример, dupli Finder - графическая программа под Windows и Linux. Такой же поиск может работать и в интернете. С помощью MD5 проверяют целостность скачанных файлов - так, некоторые программы идут вместе со значением хеша. Например, диски для инсталляции.MD5 используется для хеширования паролей. В системе UNIX каждый пользователь имеет свой пароль и его знает только пользователь. Для защиты паролей используется хеширование. Получить настоящий пароль можно только полным перебором. При появлении UNIX единственным способом хеширования был DES (Data Encryption Standard), но им могли пользоваться только жители США, потому что исходные коды DES нельзя было вывозить из страны. Во FreeBSD решили эту проблему. Пользователи США могли использовать библиотеку DES, а остальные пользователи имеют метод, разрешённый для экспорта. Поэтому в FreeBSD стали использовать MD5 по умолчанию. Многие системы используют базу данных для хранения паролей и существует несколько способов для хранения паролей. Пароли хранятся как есть. При взломе такой базы все пароли станут известны. Хранятся только хеши паролей (с помощью MD5, SHA). Найти пароли можно только полным перебором. Но сейчас такая задача решается за доли секунды. Пароль из таблицы был найден всего за 0,036059 сек. Хранятся хеши паролей и несколько случайных символов. К каждому паролю добавляется несколько случайных символов (их ещё называют «salt» или «соль») и результат ещё раз хешируется. Например, md5(md5(pass)+word). Найти пароль с помощью таблиц таким методом не получится.

Существует несколько надстроек над MD5.

- MD5(HMAC) - HMAC (Keyed-Hashing for Message Authentication хеширование с ключом для аутентификации сообщения) - алгоритм позволяет хешировать входное сообщение L с некоторым ключом K, такое хеширование позволяет аутентифицировать подпись;
- MD5 (Base64)- здесь полученный MD5 хеш кодируется алгоритмом Base64;
- MD5 (Unix) - алгоритм вызывает тысячу раз стандартный MD5.

Таблица 2.5

SHA-1

Название	SHA-1
Создан	1995 г
Опубликован	1995 г
Размер хэша	160 бит
Число раундов	5
Тип	Хеш-функция

Secure Hash Algorithm1 - алгоритм криптографического хеширования. Для входного сообщения произвольной длины (максимум $2^{64} - 1$ бит) алгоритм генерирует 160-битное хеш-значение, называемое также дайджестом сообщения. Используется во многих криптографических приложениях и протоколах. Также рекомендован в качестве основного для государственных учреждений в США. Принципы, положенные в основу SHA-1, аналогичны тем, которые использовались Рональдом Ривестом при проектировании MD4.

Описание алгоритма. SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш блока M_i равен $h_i = f(M_i, h_{i-1})$. Хеш-значением всего сообщения является выход последнего блока.

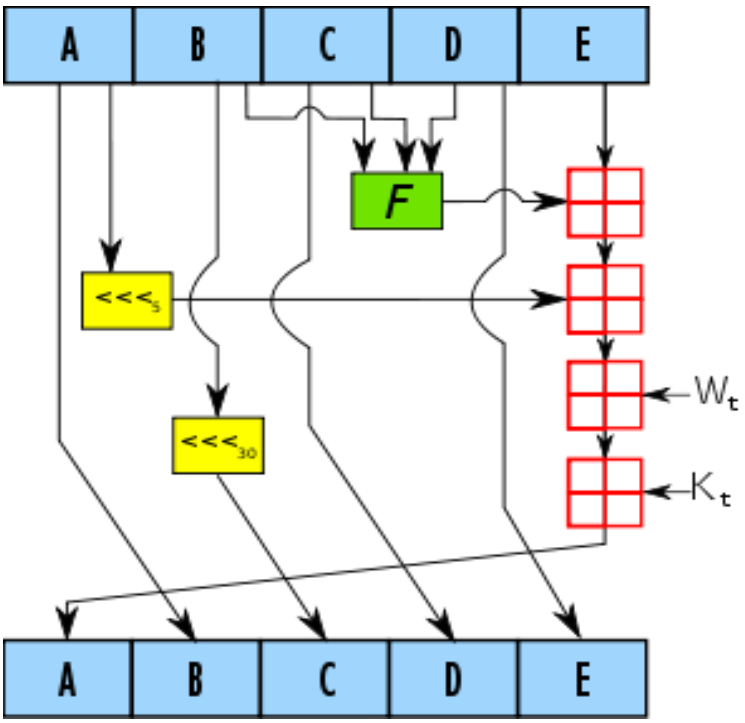


Рис.2.4. Одна итерация алгоритма SHA1

Инициализация. Исходное сообщение разбивается на блоки по 512 бит в каждом. Последний блок дополняется до длины, кратной 512 бит. Сначала добавляется 1 а потом нули, чтобы длина блока стала равной (512- 64 = 448) бит. В оставшиеся 64 бита записывается длина исходного сообщения в битах.

Если последний блок имеет длину более 448, но менее 512 бит, дополнение выполняется следующим образом: сначала добавляется 1, затем нули вплоть до конца 512-битного блока; после этого создается ещё один 512-битный блок, который заполняется вплоть до 448 бит нулями, после чего в оставшиеся 64 бита записывается длина исходного сообщения в битах. Дополнение последнего блока осуществляется всегда, даже если сообщение уже имеет нужную длину.

Инициализируются пять 32-битовых переменных.

A = a = 0x67452301

B = b = 0xEFCDAB89

C = c = 0x98BADCFE

D = d = 0x10325476

E = e = 0xC3D2E1F0

Таблица 2.6

Определяются четыре нелинейные операции и четыре константы

$F_t(m,l,k) = (m \Lambda l) \vee (m \Lambda k)$	K_t = 0x5A827999	0≤t≤19
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0x6ED9EBA1	20≤t≤39
$F_t(m,l,k) = (m \Lambda l) \vee (m \Lambda k) \vee (l \Lambda k)$	K_t = 0x8F1BBCDC	40≤t≤59
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0xCA62C1D6	60≤t≤79

Главный цикл. Главный цикл итеративно обрабатывает каждый 512-битный блок. Итерация состоит из четырех этапов по двадцать операций в каждом[14]. Блок сообщения преобразуется из 16 32-битовых слов M_i в 80 32-битовых слов W_j по следующему правилу:

$W_t = M_t$ при 0≤t≤15

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) << 1$ при 16≤t≤79

здесь <<— это циклический сдвиг влево

для *t* **от** 0 **до** 79

temp = (a<<5) + F_t(b,c,d) + e + W_t + K_t

e = d

d = c

c = b<<30

b = a

a = temp

После этого a, b, c, d, e прибавляются к A, B, C , D , E соответственно. Начинается следующая итерация.

Итоговым значением будет объединение пяти 32-битовых слов в одно 160-битное хеш-значение.

Псевдокод SHA-1. Псевдокод алгоритма SHA-1 следующий:

Инициализация переменных:

h0 = 0x67452301

h1 = 0xEFCDAB89

h2 = 0x98BADCFE

h3 = 0x10325476

Предварительная обработка: Присоединяем бит '1' к сообщению.

Присоединяем k битов '0', где k наименьшее число ≥ 0 такое, что длина получившегося сообщения (в битах) сравнима по модулю 512 с 448 (length mod 512==448). Добавляется длину исходного сообщения (до предварительной обработки) как целое 64-битное Big-endian число, в *битах*.

Вместо оригинальной формулировки FIPS PUB 180-1 приведены следующие эквивалентные выражения и могут быть использованы на компьютере f в главном цикле:

(0 ≤ i ≤ 19): f = d **xor** (b **and** (c **xor** d)) *(альтернатива 1)*

(0 ≤ i ≤ 19): f = (b **and** c) **xor** ((**not** b) **and** d) *(альтернатива 2)*

(0 ≤ i ≤ 19): f = (b **and** c) + ((**not** b) **and** d) *(альтернатива 3)*

(40 ≤ i ≤ 59): f = (b **and** c) **or** (d **and** (b **or** c)) *(альтернатива 1)*

(40 ≤ i ≤ 59): f = (b **and** c) **or** (d **and** (b **xor** c)) *(альтернатива 2)*

(40 ≤ i ≤ 59): f = (b **and** c) + (d **and** (b **xor** c)) *(альтернатива 3)*

(40 ≤ i ≤ 59): f = (b **and** c) **xor** (b **and** d) **xor** (c **and** d) *(альтернатива 4)*

Примеры. Ниже приведены примеры хешей SHA-1. Для всех сообщений подразумевается использование кодировки ASCII.

Хеш панграммы на русском:

SHA-1(«В чащах юга жил бы цитрус? Да, но фальшивый экземпляр!»)
= 9e32295f 8225803b b6d5fdfc c0674616 a4413c1b

Хеш панграммы на английском:

SHA-1(«The quick brown fox jumps over the lazy dog»)
= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

SHA-1(«sha2)
= d8f45903 20e1343a 915b6394 170650a8 f35d6926

Небольшое изменение исходного текста (одна буква в верхнем регистре) приводит к сильному изменению самого хеша. Это происходит вследствие лавинного эффекта.

SHA-1("Sha") = ba79baeb 9f10896a 46ae7471 5271b7f5 86e74640

Даже для пустой строки вычисляется нетривиальное хеш-значение.

SHA-1("") = da39a3ee 5e6b4b0d 3255bfef 95601890 afd80709

Криптоанализ. Криптоанализ хеш-функций направлен на исследование уязвимости к различного вида атакам. Основные из них:

- нахождение коллизий - ситуация, когда двум различным исходным сообщениям соответствует одно и то же хеш-значение;
- нахождение прообраза — исходного сообщения — по его хешу.

При решении методом «грубой силы»:

- вторая задача требует 2^{160} операций;
- первая же требует в среднем $2^{160/2} = 2^{80}$ операций, если использовать парадокс дней рождения.

В частности анализ основан на оригинальной дифференциальной атаке на SHA-0, «near-collision» атаке на SHA-0, мультиблоковой методике, а также методикам модификации исходного сообщения, использованных при атаках поиска коллизий на HAVAL-128, MD4, RIPEMD и MD5.

Различия:

1. В SHA-1 на четвертом этапе используется та же функция f, что и на втором этапе.
2. В MD5 в каждом действии используется уникальная прибавляемая константа. В SHA-1 константы используются повторно для каждой из четырех групп.
3. В SHA-1 добавлена пятая переменная.
4. SHA-1 использует циклический код исправления ошибок.
5. В MD5 четыре сдвига, используемые на каждом этапе отличаются от значений, используемых на предыдущих этапах. В SHA на каждом этапе используется постоянное значение сдвига.
6. В MD5 четыре различных элементарных логических функции, в SHA-1 – 3.
7. В MD5 длина дайджеста составляет 128 бит, в SHA-1 - 160 бит.

SHA-1 содержит больше раундов (80 вместо 64) и выполняется на 160-битном буфере по сравнению со 128-битным буфером MD5. Таким образом, SHA-1 должен выполняться приблизительно на 25% медленнее, чем MD5 на той же аппаратуре.

Использование. Хеш-функции используются в системах контроля версий, системах электронной подписи, а также для построения кодов аутентификации. SHA-1 является наиболее распространенным из всего семейства SHA и применяется в различных широко распространенных криптографических приложениях и алгоритмах.

SHA-1 используется в следующих приложениях:

- S/MIME - дайджесты сообщений;
- SSL - дайджесты сообщений;
- IPSec - для алгоритма проверки целостности в соединении «точка-точка»;
- SSH - для проверки целостности переданных данных;
- PGP- - для создания электронной цифровой подписи;
- Git - для идентификации каждого объекта по SHA-1-хешу от хранимой в объекте информации.

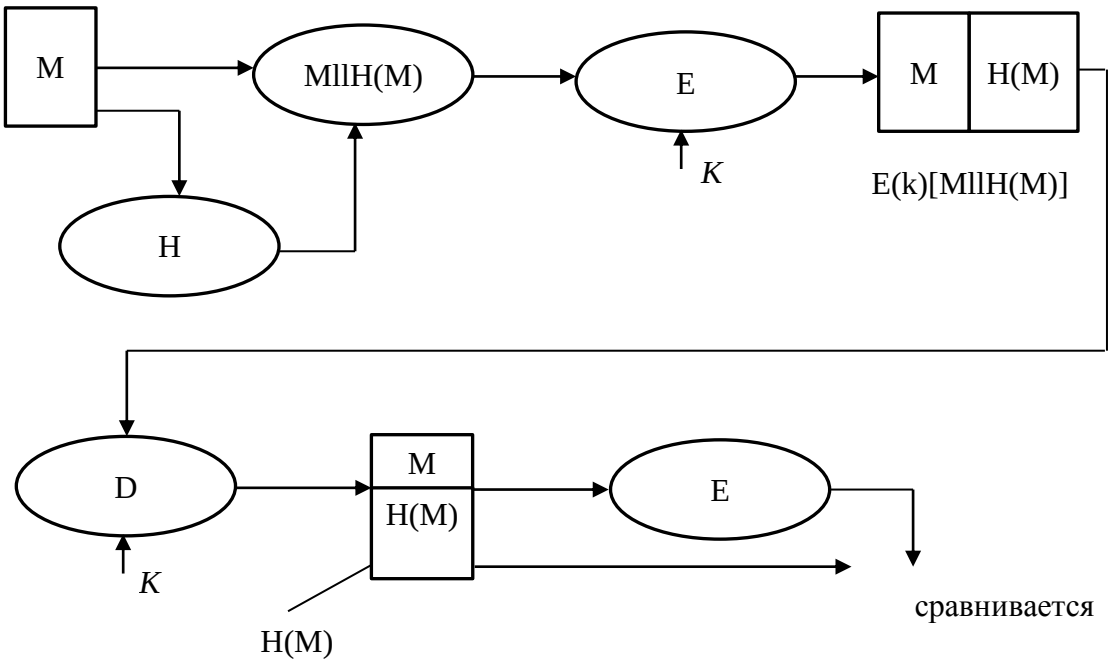


Рис.2.5(а). Схемы использование хэш-функции

а) A→B : E(k)[M||H(M)]

- обеспечивает секретности;
- обеспечивает целостности (H(M) – криптографический защищен).

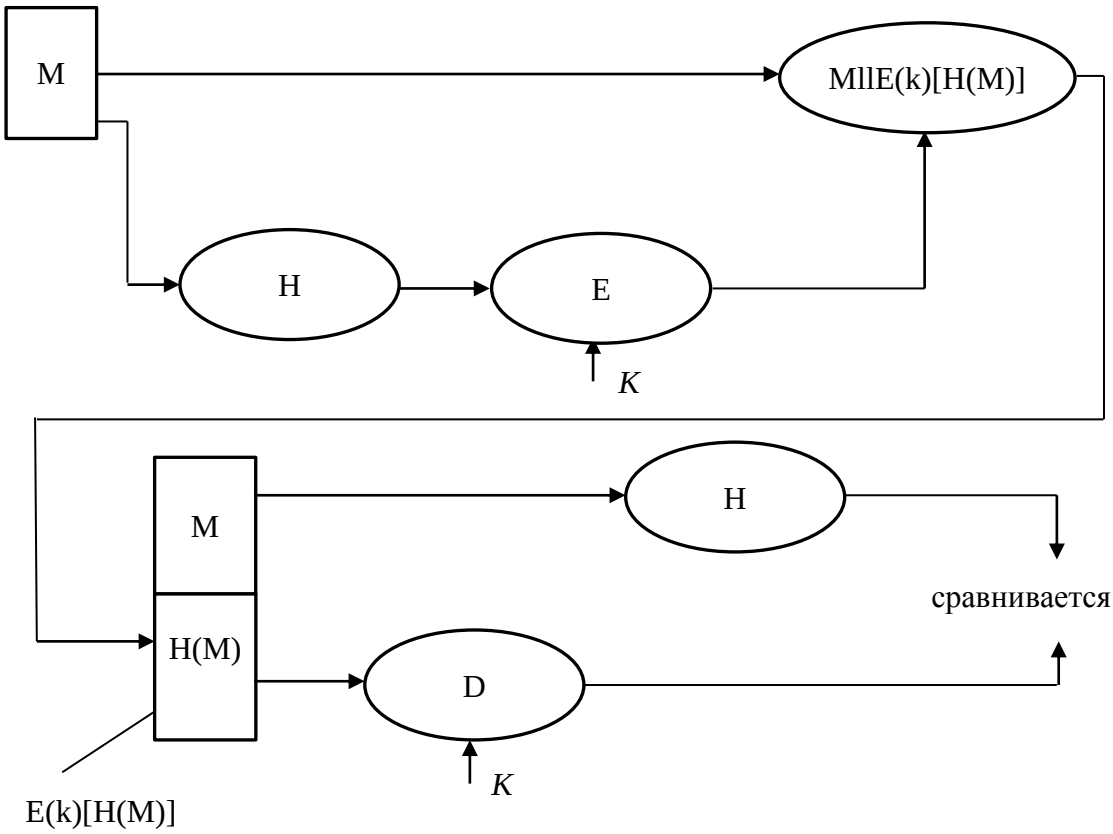


Рис.2.5(б). Схемы использование хэш-функции

б) A→B:M||E(k)[H(M)]

- обеспечивает целостности (H(M) – криптографический защищен).

2.4. Методы обеспечение целостностью информации с помощью HMAC

HMAC (сокращение от англ. hash-based message authentication code, хеш-код идентификации сообщений) - алгоритм усиления криптостойкости других криптоалгоритмов.

Алгоритм. Хеш-функция разделяет сообщения на блоки фиксированного размера и применяет к ним функцию сжатия. (MD5 или SHA-1 используют блоки 512 бит). После применения HMAC размер результата не меняется (128 или 160 бит для MD5 и SHA-1).

Функция HMAC определяется следующим образом:

$$HMAC_K(m) = h\big((K \oplus opad) \| h((K \oplus ipad) \| m)\big)$$

где:

1. h - хеш-функция
2. K - секретный ключ, дополненный нулями до размера блока
3. m - сообщение для идентификации
4. $\|$ - конкатенация
5. $\oplus$ - xor
6. opad - 0x5c5c..5c (длина равна размеру блока)
7. ipad - 0x3636..36 (длина равна размеру блока)

Если длина секретного ключа превышает размер блока, то ключ необходимо укоротить, преобразовав его с помощью функции h, и дополнить нулями до размера блока.

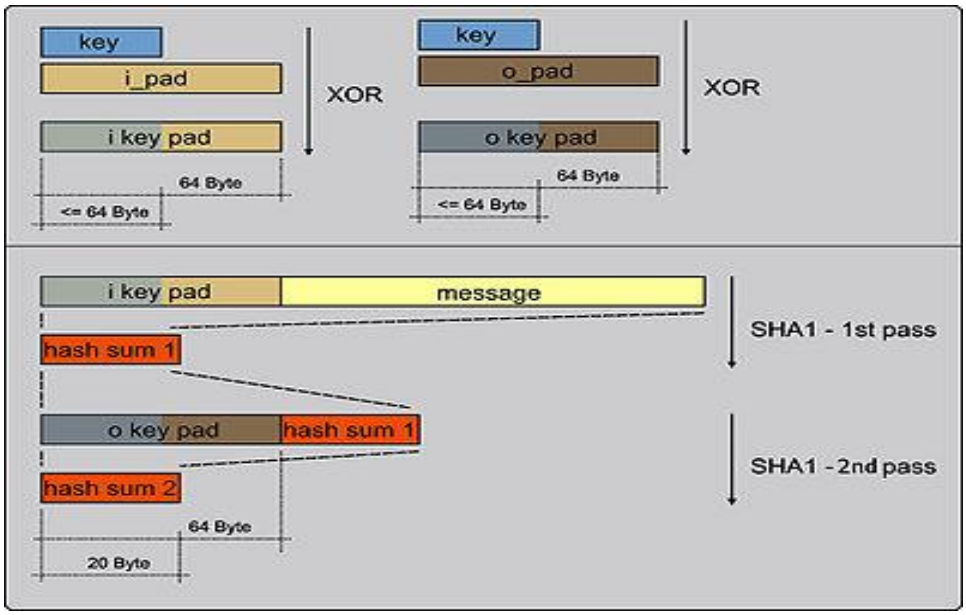


Рис.2.6. Схема HMAC

В криптографии HMAC (Хэш основе Message Authentication Code), является специфической конструкции для расчета кода аутентификации сообщений (MAC) с участием криптографической хэш-функция в сочетании с секретным ключом[15]. Как и в любой MAC, она может быть использована одновременно проверить и целостность данных и подлинность сообщения. Любое итерационных криптографической хэш-функции, как например MD5

или SHA-1, могут быть использованы при расчете HMAC; в результате MAC алгоритм называется HMAC-MD5 и HMAC-SHA1 соответственно. Криптографической стойкости HMAC зависит от криптографической стойкости основных хэш-функция, размер хэш длиной выход в битах и от размера и качества криптографического ключа.

Итерационных хэш-функция разбивает сообщение на блоки фиксированного размера и перебирает их функции сжатия. Так, например, MD5 и SHA-1 работает на 512-битные блоки. Размер выходного КСОМ то же, что и основные функции хеширования (128 или 160 бит в случае MD5 или SHA-1, соответственно), хотя он может быть усечен по желанию.

HMAC SHA 1. HMACSHA1 - это хэш-алгоритм с ключом, созданный на основе хэш-функции SHA1 и используемый для вычисления хэш-кода проверки подлинности сообщения (HMAC).Процесс вычисления кода HMAC заключается в смешивании секретного ключа с данными сообщения, вычислении хэш-функции результата, повторном смешивании хэш-значения с секретным ключом и повторном применении хэш-функции[16]. Длина выходного хэша составляет 160 бит. Код HMAC может использоваться для выявления факта подделки сообщения, передаваемого по незащищенному каналу, при условии, что секретный ключ известен отправителю и получателю. Отправитель вычисляет хэш-значение исходных данных, а затем передает исходные данные и хэш-значение в одном сообщении. Получатель повторно вычисляет значение хэша полученных данных и проверяет, совпадает ли оно с полученным кодом HMAC.

Любое изменение данных или значения хэша вызовет несовпадение, поскольку для изменения сообщения и повторного создания правильного кода HMAC необходимо знать секретный ключ. Таким образом, если исходное значение хэша совпадает с вычисленным, считается, что подлинность сообщения установлена.

3. Безопасность жизнедеятельности

3.1. Вибрация как вредный производственный фактор

Вибрация – один из наиболее распространенных неблагоприятных физических факторов окружающей среды, приобретающих важное социально-гигиеническое значение, в связи с урбанизацией, а также механизацией и автоматизацией технологических процессов, дальнейшим развитием авиации, транспорта. Шум определяют как звук, оцениваемый негативно и наносящий вред здоровью. Вибрация – это совокупность апериодических звуков различной интенсивности и частоты (шелест, дребезжание, скрип, визг и т.п.). С физиологической точки зрения шум – это всякий неблагоприятно воспринимаемый звук.

Окружающие человека шумы имеют разную интенсивность: разговорная речь – 50…60 дБ А, автосирена – 100 дБ А, шум двигателя легкового автомобиля –80 дБ А, громкая музыка –70 дБ А, шум от движения трамвая –70…80 дБ А, шум в обычной квартире –30…40 дБ А.

По спектральному составу в зависимости от преобладания звуковой энергии в соответствующем диапазоне частот различают низко-, средне-и высокочастотные вибрации. Шум по временным характеристикам может быть постоянным по уровню и непостоянным. Непостоянный шум может быть колеблющимся по уровню, прерывистым и импульсным, по длительности действия – продолжительным и кратковременным. Шум как акустический процесс характеризуется с физической и физиологических сторон. С гигиенических позиций придается большое значение амплитудно-временным, спектральным и вероятностным параметрам непостоянных шумов, наиболее характерных для современного производства. С физической стороны он представляет собой явление, связанное с волнообразным распространением колебаний частиц упругой среды.

По физической сущности шум – это волнообразное движение частиц упругой среды (газовой, жидкой или твердой) и поэтому характеризуется

амплитудой колебания (м), частотой (Гц), скоростью распространения (м/с) и длиной волны (м). Характер негативного воздействия на органы слуха и подкожный рецепторный аппарат человека зависит еще и от таких показателей вибрации, как уровень звукового давления (дБ) и громкость. Первый показатель называется силой звука (интенсивностью) и определяется звуковой энергией в эргах, передаваемой за секунду через отверстие в 1 $см^2$. Громкость вибрации определяется субъективным восприятием слухового аппарата человека. Порог слухового восприятия зависит еще и от диапазона частот. Так, ухо менее чувствительно к звукам низких частот.

В различных отраслях экономики имеются источники вибрации – это механическое оборудование, людские потоки, городской транспорт.

По источникам возникновения шум классифицируется:

1) Механический шум, обусловленный колебаниями деталей машин и их взаимным перемещением.

2) Аэрогидродинамические шумы возникают при движении газов и жидкостей, их взаимодействия с твердыми телами.

3) Электромагнитный шум возникает в электрических машинах и электрооборудовании.

По характеру спектра шум классифицируется на:

1) Широкополосный шум.

2) Тональный шум.

Звуковые колебания различных диапазонов и спектрального состава могут возникать в результате работы машин, агрегатов, вентиляторов, компрессоров, газотурбинных установок, нагревательных печей, трансформаторов и др. Автотранспортные средства: автобусы, грузовые и легковые машины, средства железнодорожного, воздушного и водного транспорта также являются источниками акустических колебаний. Воздействие вибрации на организм человека вызывает негативные изменения прежде всего в органах слуха, нервной и сердечно-сосудистой системах. Степень выраженности этих изменений зависит от параметров вибрации,

стажа работы в условиях воздействия вибрации, длительности действия вибрации в течение рабочего дня, индивидуальной чувствительности организма. Субъективная реакция человека на шумовое воздействие зависит от степени умственного и физического напряжения, возраста, пола, состояния здоровья, длительности влияния и уровня вибрации[17].

Под влиянием вибрации у людей изменяются показатели переработки информации, снижается темп и ухудшается качество выполняемой работы.

Изучение влияния вибрации на жителей разного пола и возраста показало, что более чувствительны к нему женщины и лица старших возрастных групп. Длительное воздействие вибрации на организм человека приводит к развитию утомления, нередко переходящего в переутомление, к снижению производительности и качества труда. Особенно неблагоприятно шум действует на орган слуха. Шум с уровнем звукового давления до 30…35 дБ привычен для человека и не беспокоит его. Повышение этого уровня до 40…70 дБ в условиях среды обитания создает значительную нагрузку на нервную систему, вызывая ухудшение самочувствия и при длительном действии может быть причиной неврозов. Воздействие вибрации уровнем свыше 75 дБ может привести к потере слуха – профессиональной тугоухости. При действии вибрации высоких уровней (более 140 дБ) возможен разрыв барабанных перепонок, контузия, а при еще более высоких (более 160 дБ) и смерть. Как правило, оба уха страдают в одинаковой степени. Начальные проявления профессиональной тугоухости чаще всего встречаются у лиц со стажем работы в условиях вибрации около 5 лет. Риск потери слуха у работающих при 10-летней продолжительности воздействия вибрации составляет 10% при уровне 90 дБ (шкала А), 29% – при 100 дБ (шкала А) и 55% – при 110 дБ (шкала А).

Неспецифическое воздействие вибрации обычно проявляется раньше, чем изменения в органе слуха, и выражается в нарушениях нервно-психической сферы в форме невротического и астенического синдрома в сочетании с вегетативной дисфункцией, сопровождающихся

раздражительностью, общей слабостью, головной болью, головокружением, повышенной утомляемостью, расстройством сна, ослаблением памяти и др. Не исключена возможность развития нейроциркуляторного синдрома, преимущественно по гипертоническому типу. Больные с потерей слуха требуют рационального трудоустройства, переквалификации или переводятся на инвалидность. Прием на работу с поражением органов слуха и гипертонической болезнью исключен.

Механизм комплексного действия вибрации на организм сложен и недостаточно изучен. Наряду с органом слуха восприятие звуковых колебаний часто может осуществляться и через кожный покров рецепторами вибрационной чувствительности. Это подтверждается наблюдениями о том, что люди, лишенные слуха, при прикосновении к источникам, генерирующим звуки, не только ощущают их, но и могут оценивать звуковые сигналы определенного характера.

Симптомы снижения слуха, которое бывает обычно двусторонним: звон в ушах, головная боль, быстрая утомляемость, нарушения сна, боли в сердце. В производственных условиях воздействие вибрации на работающих обычно сочетается с рядом других неблагоприятных факторов – вибрацией, определенной степенью напряженности и тяжести труда, неудовлетворительными микроклиматическими условиями, воздействием химических веществ, инфразвука и ультразвука, электромагнитного поля и др. Шум может усугублять неблагоприятное воздействие сопутствующих факторов физической и химической природы, оказывая прежде всего отрицательное влияние на состояние здоровья и работоспособность профессиональных групп, труд которых сопровождается нервным напряжением.

У лиц, подвергающихся воздействию вибрации, также могут наблюдаться изменения секреторной и моторной функций желудочно-кишечного тракта, сдвиги в обменных процессах – нарушение основного, витаминного, углеводного, белкового, жирового и солевого обменов,

нарушения функционального состояния сердечно-сосудистой системы в виде брадикардии, повышения тонуса периферических сосудов и др. Помимо действия вибрации на органы слуха установлено его вредное влияние на многие органы и системы организма, в первую очередь на центральную нервную систему, функциональные изменения в которой происходят раньше, чем диагностируется нарушение слуховой чувствительности. Поражение нервной системы под действием вибрации сопровождается раздражительностью, ослаблением памяти, апатией, подавленным настроением, изменением кожной чувствительности и другими нарушениями, в частности замедляется скорость психических реакций, наступает расстройство сна и т.д. У работников умственного труда происходит снижение темпа работы, ее качества и производительности.

Интенсивный шум на производстве способствует снижению внимания и увеличению числа ошибок при выполнении работы, исключительно сильное влияние оказывает шум на быстроту реакции, сбор информации и аналитические процессы, из-за вибрации снижается производительность труда и ухудшается качество работы. Шум затрудняет своевременную реакцию работающих на предупредительные сигналы внутрицехового транспорта (автопогрузчиков, мостовых кранов и т.п.), что способствует возникновению несчастных случаев на производстве.

Таким образом, воздействие вибрации может привести к сочетанию профессиональной тугоухости (неврит слухового нерва) с функциональными расстройствами центральной нервной, вегетативной, сердечно-сосудистой и других систем, которые могут рассматриваться как профессиональное заболевание – шумовая болезнь. Профессиональный неврит слухового нерва (шумовая болезнь) чаще всего встречается у рабочих различных отраслей машиностроения, текстильной промышленности и проч. Случаи заболевания встречаются у лиц, работающих на ткацких станках, с рубильными, клепальными молотками, обслуживающих прессоштамповочное оборудование, у испытателей-мотористов и других профессиональных групп,

длительно подвергающихся интенсивному шуму. Для нормирования постоянных шумов применяют допустимые уровни звукового давления в 9 октавных полосах частот в зависимости от вида деятельности. Для ориентировочной оценки в качестве характеристики постоянного широкополосного вибрации на рабочих местах допускается принимать уровень звука (дБ А), определяемый по шкале А шумомера с коррекцией низкочастотной составляющей по закону чувствительности органов слуха и приближением результатов объективных измерений к субъективному восприятию. Защита от вибрации достигается разработкой шумобезопасной техники, применением средств и методов коллективной защиты, индивидуальной защиты и строительно-акустическими методами. Акустические средства защиты от вибрации в зависимости от принципа действия подразделяются на средства звукоизоляции, средства звукопоглощения, глушители вибрации. Наиболее эффективным методом защиты является уменьшение вибрации в источнике его образования, что достигается применением технологических процессов и оборудования, не создающих чрезмерного вибрации. Наиболее эффективными звукоизолирующими материалами являются:

1. Трипласт.
2. Полимерные покрытия.
3. Пластобетон.

Звукопоглощающие материалы: мрамор, бетон, гранит, кирпич, фанера, стекловата.

Для борьбы с шумом в помещениях проводятся мероприятия как технического, так и медицинского характера. Основными из них являются:

- устранение причины вибрации, т.е. замена шумящего оборудования, механизмов на более современное нешумящее оборудование;
- изоляция источника вибрации от окружающей среды (применение глушителей, экранов, звукопоглощающих строительных материалов);
- ограждение шумящих производств зонами зеленых насаждений;

- применение рациональной планировки помещений;
- использование дистанционного управления при эксплуатации шумящего оборудования и машин;
- использование средств автоматики для управления и контроля технологическими производственными процессами;
- использование проивошумов – индивидуальных средств защиты.

Эти приспособления снижают уровень громкости вибрации, но не мешают восприятию необходимых команд и сигналов. Противошумы по назначению и конструктивному исполнению подразделяют на три типа: вкладыши, наушники и шлемы. Выбор индивидуальных средств защиты органа слуха зависит от мощности шумов, спектрального их состава, времени действия за рабочую смену. Противошумы следует применять с первого дня пребывания в шумной обстановке, что способствует предотвращению нарушений слуха и возникновению других неблагоприятных эффектов, связанных с воздействием вибрации:

- проведение периодических медицинских осмотров с прохождением аудиометрии;
- соблюдение режима труда и отдыха;
- проведение профилактических мероприятий, направленных на восстановление здоровья. Выполнение установленных норм и правил контролируют органы санитарной службы и общественного контроля.

Таким образом, эффективная защита работающих от неблагоприятного влияния вибрации требует осуществления комплекса организационных, технических и медицинских мер на этапах проектирования, строительства и эксплуатации производственных предприятий, машин и оборудования. В целях повышения эффективности борьбы с шумом введены обязательный гигиенический контроль объектов, генерирующих шум, регистрация физических факторов, оказывающих вредное воздействие на окружающую среду и отрицательно влияющих на здоровье людей.

3.2. Пожарная безопасность

Пожарная безопасность при эксплуатации Wi-Fi модулей на предприятиях должна обеспечиваться:

- правильным выбором степени защиты оборудования; защитой электрических аппаратов и проводников от токов короткого замыкания;
- заземлением электроприемников;
- соответствующей конструкцией модулей;
- выбором сечения проводников по безопасному нагреву, а также соблюдением противопожарных требований при канализации электроэнергии;
- надежностью электроснабжения противопожарных устройств;
- организационно-техническими мероприятиями (профилактические ремонты, испытания, обслуживание и т.п.).

Руководитель предприятия, в целях обеспечения пожарной безопасности обязан установить порядок введения в эксплуатацию Wi-Fi модулей после монтажа, планово-предупредительного и других ремонтов и испытаний, а также назначить лицо, ответственное за обеспечение пожарной безопасности при эксплуатации Wi-Fi модулей. К монтажу и эксплуатации допускаются модули, которые по своему типу и исполнению соответствует классу пожароопасной, взрывоопасной зоны, а также характеристике окружающей среды. Запрещается эксплуатировать в пожароопасных и взрывоопасных зонах модули, изготовленные неспециализированными организациями, а также не имеющие паспорта или инструкции по эксплуатации.

Устройства проходов кабелей или трубопроводов сквозь стены, перекрытия и переходы через температурные и усадочные швы в пожароопасных и взрывоопасных зонах должны содержаться в исправном состоянии и обеспечивать надежную защиту от распространения огня в смежные помещения. Все электроустановки и Wi-Fi модули должны быть

обеспечены аппаратами защиты от токов короткого замыкания и других ненормальных режимов работы. Плавкие вставки предохранителей должны быть калиброваны с указанием на клейме номинального тока вставки. Использовать самодельные и нестандартные плавкие вставки аппаратов защиты не допускается. Периодически должен производиться замер сопротивления изоляции проводов и кабелей. Запрещается эксплуатировать провода и кабели, сопротивление изоляции которых не соответствует требованиям нормативных документов. Расстояние от светильников и других электрических установок до сгораемых материалов должно быть не менее 0,5 м. Wi-Fi модули необходимо периодически очищать от горючей пыли или отложений, не допуская их накопления. Периодичность очистки должна устанавливаться в инструкциях о мерах пожарной безопасности. После окончания работы все электроустановки и Wi-Fi модули в помещениях, за исключением специального назначения, необходимо отключать.

При эксплуатации Wi-Fi модулей запрещается:

- использовать модули, поверхностный нагрев которых при работе превышает температуру окружающей среды более чем на 40 °С, если к ним не предъявлены иные требования;
- пользоваться кабелями и проводами с поврежденной изоляцией, а также потерявшей в процессе эксплуатации защитные электроизоляционные свойства;
- оставлять под напряжением провода и кабели с неизолированными концами, а также неиспользуемые электрические сети;
- пользоваться поврежденными или неисправными розетками, распределительными коробками, рубильниками, защитными устройствами и другими электроустановочными изделиями;
- оклеивать и окрашивать электропровода, завязывать их в узлы, подвешивать непосредственно на провода светильники, установочную аппаратуру и другие предметы;

- включать Wi-Fi модули, автоматически отключившиеся при коротком замыкании или токах перегрузки, без выяснения и устранения причин отключения;
- включать Wi-Fi модули, не обеспеченные аппаратами защиты;
- перегружать провода и кабели сверх номинальных параметров;
- менять защиту (тепловые элементы, предохранители и др.) модулей другими видами защиты или защитой с другими номинальными параметрами, на которые данное оборудование не рассчитано;
- прокладывать электропровода и кабели непосредственно внутри сгораемых конструкций и под сгораемыми отделочными материалами.

Так как общие сведения о пожарной безопасности представляют непосредственный интерес при организации работы на предприятии, не лишним было бы рассмотреть и её. Причины возникновения пожаров – неправильное устройство или неисправное состояние нагревательных приборов, электроустановок, неосторожное обращение с легковоспламеняющимися жидкостями, нарушение правил поведения сварочных работ, нарушение пожарного режима (неосторожное обращение с огнем, курение в недозволенных местах). Многие пожары возникают от самовозгорания веществ, действия статического электричества и д.р.

Проектирование, строительство и эксплуатация зданий и сооружений, а так же разработок технологических процессов ведутся таким образом, чтобы вероятность возникновения пожара и вероятность воздействия опасных факторов пожара на людей не превышала 10-6 в год. Установки пожаротушения подразделяются на ручные и автоматические. Сегодня мы бы хотели рассказать об автоматических установках, отличительной особенностью которых является одновременное выполнение ими функций пожарной сигнализации, то есть обнаружения возгорания. Мы проведем их классификацию и обсудим преимущества и недостатки каждого вида. Общая схема классификации автоматических установок пожаротушения представлена на рис.3.1.

Рис.3.1. Классификация автоматических установок пожаротушения

Соответственно, полное наименование автоматической установки пожаротушения должно звучать примерно так: «Модульная система порошкового пожаротушения по площади с автоматическим пуском».

1. Классификация по типу огнетушащего вещества.

2. Водяные и пенные установки пожаротушения.

3. Вода – одно из самых распространенных веществ, используемых для тушения возгораний[18].

По типу оросителей, установки водяного и пенного пожаротушения подразделяются на спринклерные и дренчерные. Дренчер представляет себе обычный ороситель направленного действия. Отличие же конструкции спринклера состоит в том, что он содержит стеклянную колбу, препятствующую выходу огнетушащего вещества и содержащую особую спиртовую смесь. При нагреве колбы до определенного предела, она разрушается за счет расширения спиртовой смеси и открывает доступ воде или пене к месту возгорания. Кроме того, спринклерные установки могут подразделяться по типу заполнения питающих и распределительных трубопроводов на водозаполненные, воздушные и водовоздушные. Применение воздушных и водовоздушных систем обусловлено возможной пониженной температурой в месте прокладки трубопроводов и предотвращением образования льда внутри них.

Пенные установки пожаротушения с применением пены низкой и средней кратности отличаются от водяных наличием пенообразователя, подключаемого к трубопроводной системе. Существует несколько типов пенообразователей:

- насосы-дозаторы, обеспечивающие подачу пенообразователя в трубопровод;
- автоматические дозаторы с трубой Вентури и диафрагменно-плунжерным регулятором (при увеличении расхода воды возрастает перепад давления в трубе Вентури, регулятор обеспечивает подачу дополнительного количества пенообразователя);
- пеносмесители эжекторного типа;
- баки-дозаторы, использующие перепад давления, создаваемого трубой Вентури.

К преимуществам водяных установок пожаротушения можно отнести безопасность для людей, наличие неограниченного запаса огнетушащего вещества (в случае соответствующего подключения системы к внешним водопроводным сетям). Основной недостаток водяных и пенных систем – громоздкость конструкций и оборудования, необходимость проведения сварочных работ при монтаже трубопроводов, а также проектирования и монтаж повысительной насосной станции, в случае отсутствия необходимого давления на вводе установки. Однако при применении на больших объектах, как многофункциональные центры и гипермаркеты, а также в зданиях с постоянным пребыванием большого количества людей, этот тип пожаротушения становится практически безальтернативным.

Заключение

Основные результаты работы могут быть сформулированы в следующем виде:

1. Рассмотрены угрозы в компьютерных сетях и методов обеспечение целостности информации.

2. Отражен интегрированный подход, обеспечивающий целостность информации в сетях передачи данных.

3. Исследовано модели контроля целостности данных, определяющей категории объектов данных и классов операций над защитой целостности информации.

4. Были анализированы правил, определяющих взаимоотношения элементов данных и процедур в процессе функционирования системы и описаны особенности обеспечение целостности информации в компьютерных сетях.

5. Проведены анализа избыточные и CRC коды, позволяющей обнаруживать и исправлять одиночные ошибки целостности информации.

6. Формализован алгоритм расчёта CRC16 для получения контрольной суммы полиномов и выбраны алгоритмы хэш-функции и их функциональные роли в обеспечения целостности информации в сетях передачи данных.

7. Исследованы методов обеспечение целостностью информации с помощью HMAC.

8. Предложена программа, позволяющая шифровать и дешифровать информации на основе алгоритм RC6.

Список использованной литературы

1. Постановление Президента Республики Узбекистан «О дополнительных мерах по дальнейшему развитию информационно-коммуникационных технологий». от 21 марта 2012 года, ПП – 1730.
2. Ю.В. Романец, П.А. Тимофеев. Защита информации в компьютерных системах и сетях. 2 изд. “ Радио и Связ ” 2009—228 с.
3. В. Ф. Шаньгин .Комплексная защита информации в корпоративных системах, Москва ИД «ФОРУМ» - ИНФРА-М, 2010—421 с.
4. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты. — К.: ООО “ДС”, 2008. — 688 с.
5. Б. Шнайер. Прикладная криптография 2-ое издание, 2007—223 с.
6. В.А.Семяненко. Информационная безопасность. Учебное пособие, 3-е издание Москва 2008—321 с.
7. Методы и средства защиты информации. В 2-х томах / Ленков С.В., Перегудов Д.А., Хорошко В.А., Под ред. В.А. Хорошко. - К.: Арий, 2008. - Том I. Несанкционированное получение информации. - 464 с, ил.
8. Сидельников В.М. Криптография и теория кодирования // Материалы конференции «Московский университет и развитие криптографии в России», МГУ. - 2002. - 22 с.
9. Брюс Шнайер - Секреты и ложь. Безопасность данных в цифровом мире 2003 — 373 с.
10. Желников В. Криптография от папируса до компьютера. Москва. ABF. 2007—168 с.
11. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии: Учебное пособие, 2-е изд., испр. И доп. – М.:Гелиос АРВ, 2002. – 480с., ил.
12. Харин Ю.С. Берник В.И, Матвеев Г.В., Агиевич С В. Математические и компьютерные основы криптологии. - Минск: Новое знание, 2003. - 382 с.

13. Rivest RL. The MD5 message digest algorithm // Advances in Cryptology – CRYPTO’90. LNCS. Springer-Verlag. 2009. V.537. P.303-311.

14. Мао В. Современная криптография. Теория и практика. - М.: Вильяме, 2005. - 768 с.

15. Коллизии хеш-функций MD5, SHA-0,1 (Collisions in hash functions MD5, SHA-0,1).

16. www.hmac.com

17. Экология и безопасность жизнедеятельности: Учебное пособие для студентов ВУЗов / ред. Л. А. Муравий, 2002.

18. Белов С.В. Безопасность жизнедеятельности М.: Высшая школа. 2003.

Приложение

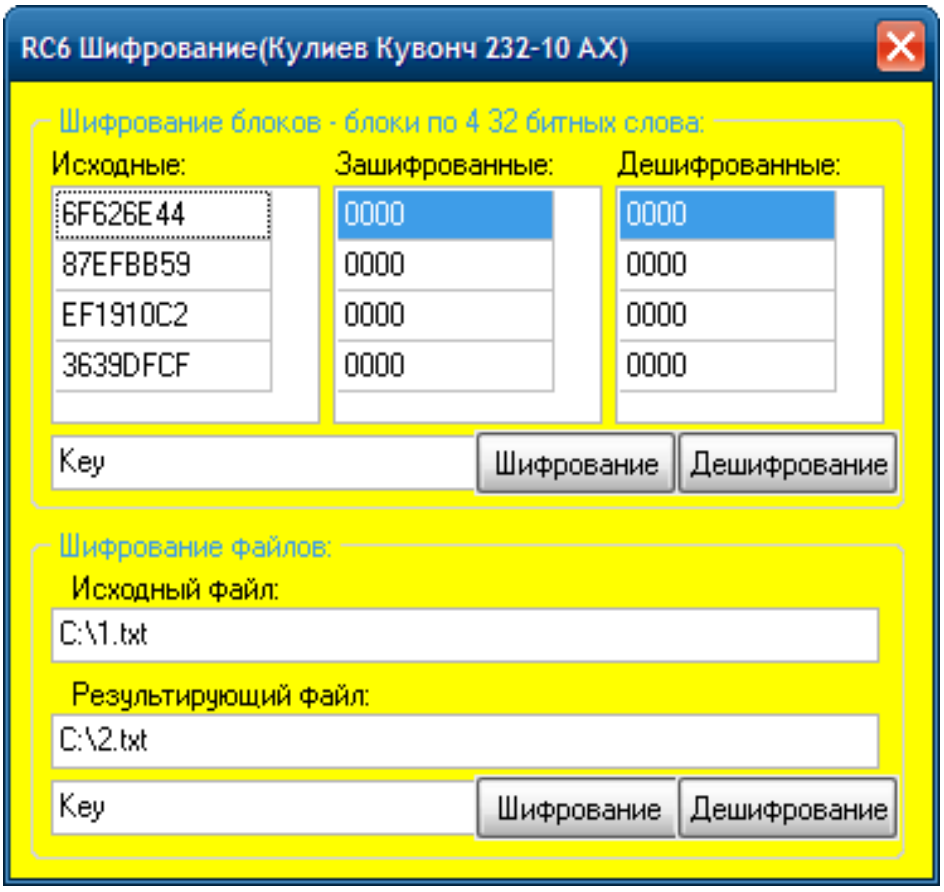
```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Grids, RC6, ExtCtrls, XPMan;
type
TForm1 = class(TForm)
GroupBox1: TGroupBox;
StringGrid1: TStringGrid;
StringGrid2: TStringGrid;
StringGrid3: TStringGrid;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Edit1: TEdit;
Button1: TButton;
Button2: TButton;
GroupBox2: TGroupBox;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
Button3: TButton;
Button4: TButton;
Label4: TLabel;
Label5: TLabel;
XPManifest1: TXPManifest;
procedure FormCreate(Sender: TObject);
procedure FormShow(Sender: TObject);
```

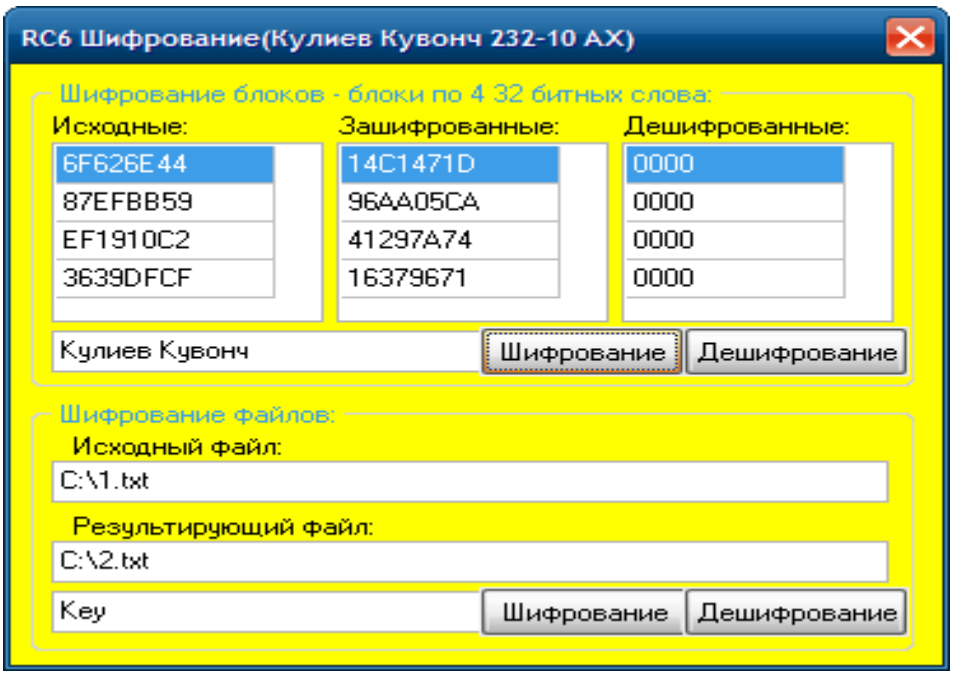
```
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
Block1,
Block2,
Block3 : TRC6Block;
implementation
{$R *.dfm}
procedure ShowResults;
var
i : Byte;
begin
for i := 1 to 4 do
begin
Form1.StringGrid1.Cells[0,i-1] := IntToHex(Block1[i], 4);
Form1.StringGrid2.Cells[0,i-1] := IntToHex(Block2[i], 4);
Form1.StringGrid3.Cells[0,i-1] := IntToHex(Block3[i], 4);
end;
end;
procedure FillBlock;
var
i : Byte;
```

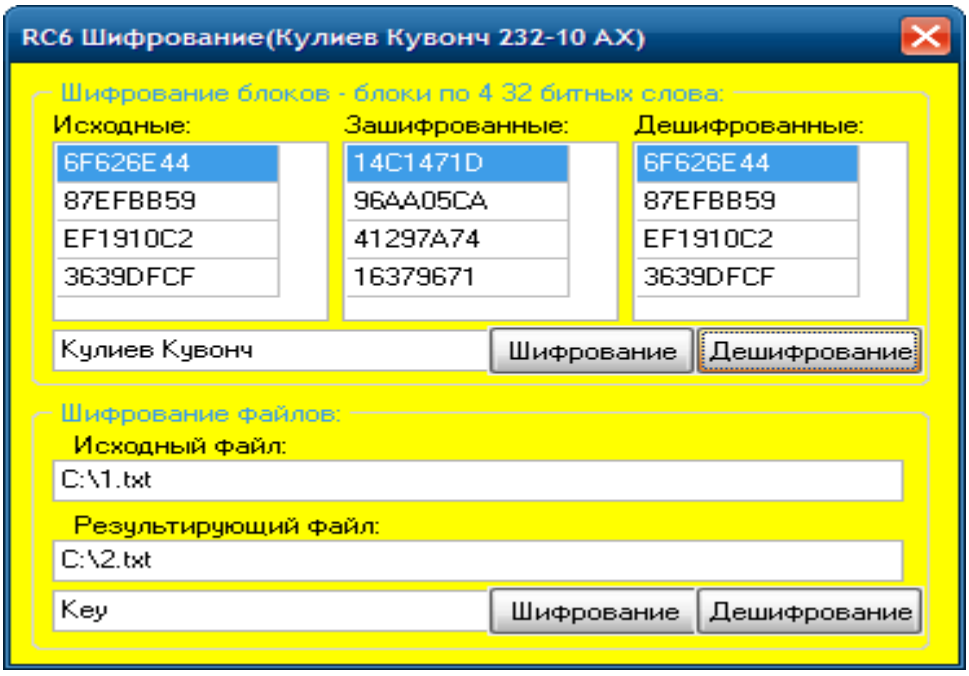
```
begin
for i := 1 to 4 do
begin
try
Block1[i] := StrToInt('$0'+Form1.StringGrid1.Cells[0,i-1]);
Block2[i] := StrToInt('$0'+Form1.StringGrid2.Cells[0,i-1]);
except
ShowMessage();
Exit;
end;
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
Randomize;
end;
procedure TForm1.FormShow(Sender: TObject);
var
i : Byte;
begin
for i := 1 to 4 do
Block1[i] := Trunc(Random(High(LongWord)));
ShowResults
end;
procedure TForm1.Button1Click(Sender: TObject);
var
i : Byte;
begin
FillBlock;
Initialize(Edit1.Text);
```

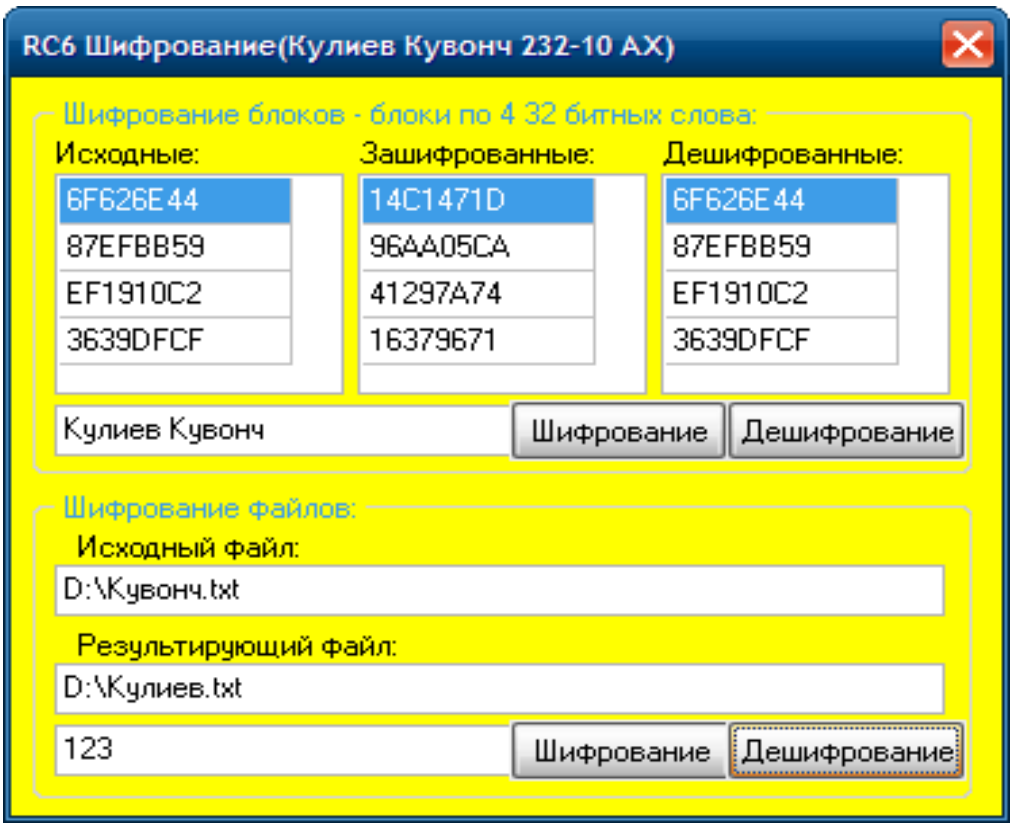
```
CalculateSubKeys;
for i := 1 to 4 do
Block2[i] := Block1[i];
EncipherBlock(Block2);
ShowResults;
end;
procedure TForm1.Button2Click(Sender: TObject);
var
i : Byte;
begin
FillBlock;
Initialize(Edit1.Text);
CalculateSubKeys;
for i := 1 to 4 do
Block3[i] := Block2[i];
DecipherBlock(Block3);
ShowResults;
end;
procedure TForm1.Button3Click(Sender: TObject);
var
Str1, Str2 : TFileStream;
begin
Str1 := TFileStream.Create(Edit2.Text, fmOpenRead);
Str2 := TFileStream.Create(Edit3.Text, fmCreate);
EncryptCopy(Str2, Str1, Str1.Size, Edit4.Text);
Str1.Free;
Str2.Free;
end;
procedure TForm1.Button4Click(Sender: TObject);
var
```

```
Str1, Str2 : TFileStream;
begin
Str1 := TFileStream.Create(Edit2.Text, fmOpenRead);
Str2 := TFileStream.Create(Edit3.Text, fmCreate);
DecryptCopy(Str2, Str1, Str1.Size, Edit4.Text);
Str1.Free;
Str2.Free;
end;
end.
```




RC6 Шифрование(Кулиев Кувонч 232-10 АХ)
Шифрование блоков - блоки по 4 32 битных слова:
Исходные:
Зашифрованные:
Дешифрованные:
6F626E44
87EFBB59
EF1910C2
3639DFCF
14C1471D
96AA05CA
41297A74
16379671
0000
0000
0000
0000
Кулиев Кувонч
Шифрование
Дешифрование
Шифрование файлов:
Исходный файл:
C:\1.txt
Результирующий файл:
C:\2.txt
Key
Шифрование
Дешифрование


RC6 Шифрование(Кулиев Кувонч 232-10 АХ)
Шифрование блоков - блоки по 4 32 битных слова:
Исходные:
Зашифрованные:
Дешифрованные:
6F626E44
87EFBB59
EF1910C2
3639DFCF
14C1471D
96AA05CA
41297A74
16379671
6F626E44
87EFBB59
EF1910C2
3639DFCF
Кулиев Кувонч
Шифрование
Дешифрование
Шифрование файлов:
Исходный файл:
C:\1.txt
Результирующий файл:
C:\2.txt
Key
Шифрование
Дешифрование


RC6 Шифрование(Кулиев Кувонч 232-10 АХ)
Шифрование блоков - блоки по 4 32 битных слова:
Исходные:
Зашифрованные:
Дешифрованные:
6F626E44
87EFBB59
EF1910C2
3639DFCF
14C1471D
96AA05CA
41297A74
16379671
6F626E44
87EFBB59
EF1910C2
3639DFCF
Кулиев Кувонч
Шифрование
Дешифрование
Шифрование файлов:
Исходный файл:
D:\Кувонч.txt
Результирующий файл:
D:\Кулиев.txt
123
Шифрование
Дешифрование