Урбанович П.П.

Краткий конспект
лекций по дисциплине «Информационная безопасность и надежность систем»

Минск, 2010

ПРЕДИСЛОВИЕ

Эффективность *информационных систем* (средств вычислительной и измерительной техники, автоматизированных систем управления, систем хранения, преобразования и передачи информации и др.) и *программных комплексов* в значительной степени определяется уровнем защищенности информации от постороннего (несанкционированного) доступа и воздействия на нее, а также достоверностью информации, которая перерабатывается в этих системах и комплексах. Указанные особенности имеют непосредственное отношение к проблемам *безопасности* и *надежности информационных систем.*

Достоверность информации зависит от целого ряда факторов как технических, обусловленных конкретной реализацией и стратегией обслуживания таких систем и устройств, так и смысловых, связанных с реализованными методами представления и преобразования информации.

Защищенность информации (каналов ее передачи и хранения) определяется простотой (или сложностью) осуществления несанкционированного доступа к носителям информации (устройствам хранения), техническим и программным средствам, реализующим операции над информацией.

В последнее время все чаще защищенность и достоверность информации рассматриваются как равнозначные аспекты двуединой *проблемы проектирования, разработки и эксплуатации безопасных и высоконадежных систем.* Такой подход базируется, прежде всего, на том, что современные информационные системы различного назначения суть *компьютерные системы* на основе *сетей* и *сетевых технологий*.

Настоящее учебно-методическое пособие предназначено для студентов, изучающих дисциплину «Информационная безопасность и надежность систем». Имеет своей целью помочь обучаемым в овладении теоретическими знаниями о методах и средствах повышения информационной безопасности и надежности систем хранения, преобразования и передачи информации, ее защиты в информационно-вычислительных системах (ИВС), приобретение практических навыков по созданию и использованию методов и средств повышения информационной безопасности и надежности систем. Пособие также может быть полезно аспирантам, изучающим избранные аспекты вышеуказанной проблемы.

В результате изучения дисциплины и выполнения заданий на лабораторных занятиях (приводятся в конце разделов) студент должен знать:

- особенности ИВС, как объекта защиты, соответствующего требуемому уровню надежности функционирования;
- организационные и правовые методы защиты информации в ИВС;
- программно-технические средства преобразования и защиты информации в ИВС, повышения надежности ИВС;
- методы криптографической защиты информации и ИВС.

Студент должен научиться применять рассматриваемые методы на практике.

1. ВВЕДЕНИЕ В ПРОБЛЕМУ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И НАДЕЖНОСТИ СИСТЕМ

1.1. Основные понятия и определения

Информация – сведения (данные)[1] о внутреннем и окружающем нас мире, событиях, процессах, явлениях и т.д., воспринимаемые и передаваемые людьми или техническими устройствами.

Информационная (информационно-вычислительная) система – организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.

Информационные процессы – процессы сбора, накопления, хранения, обработки (переработки), передачи и использования информации.

Информационные ресурсы – отдельные документы или массивы документов в информационных системах.

Доступ – специальный тип взаимодействия между объектом и субъектом, в результате которого создается поток информации от одного к другому.

Несанкционированный доступ – доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные правила доступа.

Объект – пассивный компонент системы, хранящий, перерабатывающий, передающий или принимающий информацию; примеры объектов: страницы, файлы, папки, директории, компьютерные программы, устройства (мониторы, диски, принтеры и т.д.)

Субъект – активный компонент системы , который может инициировать поток информации; примеры субъектов: пользователь, процесс либо устройство.

Безопасность ИВС – свойство системы, выражающееся в способности системы противодействовать попыткам несанкционированного доступа или нанесения ущерба владельцам и пользователям системы при различных умышленных и неумышленных воздействиях не нее.

Защита информации – организационные, правовые, программно-технические и иные меры по предотвращению угроз информационной безопасности и устранению их последствий.

Информационная безопасность систем – свойство информационной системы или реализуемого в ней процесса, характеризующее способность обеспечить необходимый уровень своей защиты.

Надежность системы – характеристика способности программного, аппаратного, аппаратно-программного средства выполнить при определенных условиях требуемые функции в течение определенного периода времени.

Достоверность работы системы (устройства) – свойство, характеризующее истинность конечного (выходного) результата работы (выполнения программы), определяемое способностью средств контроля фиксировать правильность или ошибочность работы.

---

[1] Строго говоря, в теории информации понятия «данные» и «информация» не отождествляются

Ошибка устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью или воздействующими на него помехами (преднамеренными либо непреднамеренными).

Ошибка программы – проявляется в не соответствующем реальному (требуемому) промежуточному или конечному значению (результату) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

1.2. Оценка надежности цифровых систем и каналов передачи информации

Как следует из вышеприведенного определения, *надежность* есть внутреннее свойство объекта, заложенное в него при изготовлении и проявляющееся во время эксплуатации. Вторая особенность надежности состоит в том, сто она проявляется во времени. И третья особенность – надежность проявляется по-разному при различных условиях эксплуатации и различных режимах применения *объекта* (технического объекта).

Надежность является комплексным свойством, включающим в себя единичные свойства: *безотказность*, *ремонтопригодность*, *сохраняемость*, *долговечность*.

Безотказность – это свойство технического объекта непрерывно сохранять работоспособное состояние в течение некоторого времени (или *наработки*). Наработка, как правило, измеряется в единицах времени.

Ремонтопригодность – это свойство технического объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания, ремонта (или с помощью дополнительны, избыточных технических средств, функционирующих параллельно с объектом). Большинство современных цифровых систем и устройств (в том числе, компьютеры и компьютерные системы, отдельные блоки и модули компьютеров - полупроводниковая, магнитная или оптическая память) содержат специальные средства, призванные автоматически восстанавливать работоспособность этих объектов при нарушении нормального функционирования.

Такие специальные средства контроля называются *избыточными*. На рисунке 1.1 приведена упрощенная структурная схема ИВС с избыточными средствами аппаратного контроля.

Изначальной причиной нарушения нормальной работы цифрового устройства являются технические *дефекты* (неисправности), возникающие внутри узлов или блоков устройства либо в линиях связи между ними.

Дефекты или неисправности могут приводить либо к кратковременному нарушению достоверности работы устройства (*сбой*), либо к полной и окончательной потере достоверности (*отказ*). В каждом из этих случаев следствием неисправности являются ошибки в информации (*информационные*

ошибки). Количество таких ошибок (количество ошибочных двоичных символов) принято называть *кратностью ошибки*.

Пример1-1. Исходная (правильная) информационная последовательность X_k = 1001. Длина этой последовательности равна 4 битам (k=4). Некоторая из перечисленных причин привела к тому, что в этой последовательности появились две ошибки (кратность ошибки равна двум): Y_k = 1111.

Устройства контроля и коррекции являются избыточными и предназначены соответственно для контроля за появлением ошибок и коррекции (исправления) этих ошибок.

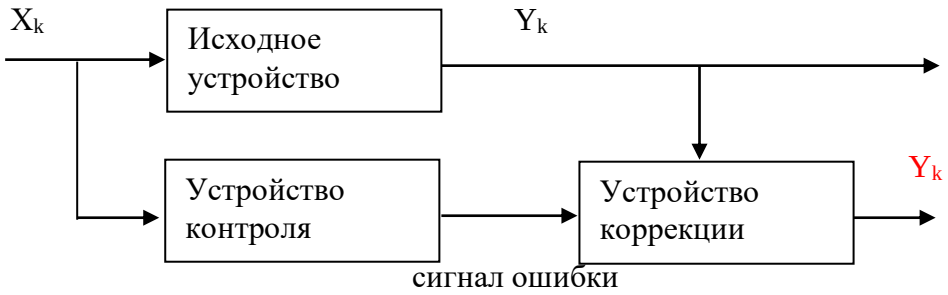


Рис.1.1. Структурная схема ИВС со средствами аппаратного контроля достоверности функционирования и коррекции выходной информационной последовательности

По возможности восстановления работоспособности после отказа различают *невосстанавливаемые* и *восстанавливаемые* объекты. В контексте дальнейшего анализа *невосстанавлиемым* будем считать цифровое устройство без избыточных схем контроля и наоборот: объект с контролем функционирования относим к классу *восстанавливаемых*.

Третье свойство надежности – *сохраняемость* – характеризует способность технического объекта сохранять в заданных пределах значения параметров после его хранения или транспортировки

Долговечность – это свойство технического объекта сохранять в заданных пределах работоспособное состояние до наступления предельного состояния при установленной системе технического обслуживания или ремонта.

Возможные состояния анализируемой ИВС (рис.1.1) можно охарактеризовать следующим образом:

1) исходное устройство действительно работает правильно ($X_k = Y_k$) в течение периода времени t; вероятность такого события обозначим $P_{пр}(t)$,
2) исходное устройство работает с ошибкой ($X_k \neq Y_k$), о чем свидетельствует сигнал ошибки; вероятность события (правильное обнаружение) – $P_{по}(t)$,
3) исходное устройство работает неправильно, однако это состояние устройством контроля не обнаруживается (пропуск, необнаружение ошибки); соответствующая вероятность – $P_{но}(t)$,
4) исходное устройство работает правильно, однако устройство контроля выдает информацию об ошибке (состояние ложной тревоги), причиной чего может

быть недостоверное функционирование самого устройства контроля; вероятность такого события обозначим $P_{лт}(t)$.

Все перечисленные события образуют полную группу событий, описываемую следующим вероятностным соотношением:

$$P_{пр}(t) + P_{по}(t) + P_{но}(t) + P_{лт}(t) = 1. \quad (1.1)$$

Надежность недостаточно определить на качественном уровне (высокая, низкая) – необходимо уметь оценивать ее качественно. Приведенные количественные параметры (вероятности) являются косвенными показателями надежности цифрового устройства.

Основным количественным показателем надежности (имеет отношение, прежде всего, к невосстанавливаемым объектам) является *наработка до первого отказа*, T_0. Вероятностные характеристики наработки и являются показателями безотказности объекта. Рассмотрим некоторые из них.

Вероятность безотказной работы, P(t). Вероятностью безотказной работы называют вероятность того, что изделие (система, устройство) будет работоспособно в течение заданной наработки при заданных условиях эксплуатации:

$$P(t) = P(T_0 > t).$$

Данная вероятность может быть рассчитана на основе статистических данных. Если принять, что N_c соответствует суммарному числу изделий (объектов), из которых N_o за время наблюдения t отказали (стали дефектными), то при достаточно большом числе N_c вероятность может быть определена как

$$P(t) = (N_c - N_o)/ N_c. \quad (1.2)$$

Данная вероятность соответствует вероятности $P_{пр}(t)$, которую мы упоминали выше.

Пример1-2. Из партии постоянно функционирующих компьютеров в количестве 1000 штук за время эксплуатации t в 6 изделиях возникли отказы. Тогда вероятность безотказной работы произвольного компьютера того же производителя при неизменном техпроцессе изготовления составляет в соответствии с (1.2):

$$P(t) = (N_c - N_o)/ N_c = (1000 – 6)/1000 = 0{,}994.$$

Пример1-3. В течение фиксированного времени (например, t=1 час) по каналам связи осуществлялась передача двоичной информации между двумя компьютерами со скоростью S = 10 Кбит/с. За время передачи 1000 символов приняты с ошибками. Определить вероятность того, что произвольный двоичный символ при передаче по тому же каналу будет принят правильно.

Нетрудно обнаружить аналогию в двух последних задачах. Если принять, что $N_c = S*t = 10\,000$ бит/с $* 3\,600$ с $= 36*10^6$ бит, а $N_o = 1000$, то искомая вероятность вычисляется как $(36*10^6 – 1000)/ 36*10^6 = 35\,999\,000/36\,000\,000 = 0{,}99997$.

Вероятность отказа Q(t) есть вероятность того, что при заданных условиях эксплуатации в течение заданной наработки произойдет хотя бы один отказ, то есть

$$Q(t) = P(T_0 < t).$$

Отказ и безотказная работа – противоположные события. Поэтому

$Q(t) = 1 - P(t)$. (1.3)

С другой стороны, следуя рассуждениям, на основе которых записано соотношение (1.2), можем также утверждать, что

$Q(t) = N_{o)}/ N_c$. (1.4)

Вероятность Q(t) в некоторых случаях соответствует вероятности $P_{по}(t)$.

Пример1-4. Из условий примера 1-2 определить вероятность отказа произвольного компьютера за время t. Как следует из (1.3) и из (1.4), искомая величина равна 0,006

Пример1-5. Из условий примера 1-3 определить вероятность приема бита с ошибкой. Легко установить, что эта величина составляет $2{,}78*10^{-5}$.

Интенсивность отказов, $\lambda(t)$ есть плотность распределения наработки до первого отказа при условии, что отказавший объект до рассматриваемого момента времени работал безотказно. Согласно вероятностному определению

$$\lambda(t) = - \ln P(t) \text{ и } P(t) = - \exp(\int_0^t \lambda(x)dx.).$$

По статистическому определению интенсивность отказов есть отношение числа отказавших в единицу времени объектов наблюдения к среднему числу работоспособных на рассматриваемом отрезке времени объектов.

Если за такой отрезок времени принять 1 час, то по условиям примера 1-3 получаем $\lambda(t) = 1000$ ч$^{-1}$.

Как видим, между тремя рассмотренными количественными характеристиками надежности (P(t), Q(t), $\lambda(t)$) существует однозначная связь. Достаточно задать одну из них, чтобы определить остальные.

1.3. Методы повышения аппаратной надежности

Вопросам обеспечения надежности уделяется внимание на всех этапах *жизненного цикла* (проектирование, изготовление, эксплуатация) устройств и каналов передачи информации. Мировой опыт показывает, что значительный эффект при решении задач обеспечения качества и надежности дает системная организация работ на основе внедрения международных стандартов серии ISO9000 или *TQM (Total Quality Management),* разработанных Международной организацией по стандартизации (*International Standard Organizatin, ISO*).

Даже краткое перечисление основных направлений работ по обеспечению надежности показывает, что современная техника обладает широким арсеналом методов повышения надежности. Эти методы можно условно разделить на следующие группы: 1) *уменьшение наработки*, 2) *снижение интенсивности отказов*, 3) *улучшение восстанавливаемости*, 4) *резервирование*. Методы первой группы основываются на использовании более быстродействующих объектов. Снижение интенсивности отказов (а также сбоев) достигается, в основном, технологией изготовления (собственно, как и уменьшение наработки). Нас интересуют остальные группы методов, поскольку к их реализации может иметь непосредственное отношение проектировщик (и пользователь) всей системы, а не

отдельных ее блоков. Улучшение восстанавливаемости (нормального функционирования) может быть достигнуто, в том числе, использованием параллельных средств контроля (см. рис.1.1). Вместе с тем, такие избыточные средства контроля можно отнести к методам, основанным на использовании резервирования. С такой точки зрения можно третью и четвертую группы объединить под единым названием – *избыточных методов*.

Будем различать *структурную, временную, информационную избыточность* либо их комбинации.

Простая структурная избыточность нередко называется *структурным резервированием*. Методы структурной избыточности подразумевают повышение надежности аппаратуры или каналов передачи за счет использования дополнительных (резервных) модулей или каналов. Характерной особенностью систем со структурным резервированием является то, что в идеально надежной системе все резервные элементы могут быть удалены без какого-либо ухудшения качества системы. Они необходимы только тогда, когда появляется принципиальная возможность появления отказа элементов системы. И соответствии с «уровнем этой принципиальной возможности», назначением системы и критичностью (стоимостью) обрабатываемой информации структурный резерв может быть «холодным» или «горячим».

Структурная схема системы с «холодным» (ждущим) резервом может быть получена изменением функций системы, представленной выше на рисунке 1.1. Такая видоизмененная структура показана на рисунке 1.2.

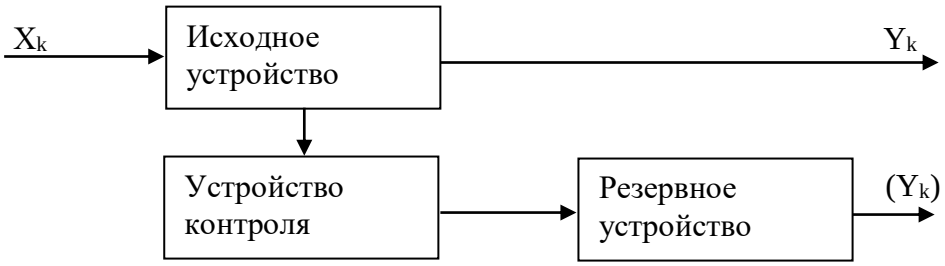


Рис.1.2. Обобщенная структурная схема системы с «холодным» резервом

Резервное устройство (канал) начинает выполнять функции основного (исходного) после установления устройством контроля отказа основном, приведшего к ошибкам в обрабатываемой информации. Надежность всей системы определяется интенсивностями отказов не только основного (исходного) устройства ($\lambda_{оу}(t)$), но и дополнительных (устройства контроля и резервного устройства): $\lambda_{ук}(t)$, $\lambda_{ру}(t)$. Понятно, что вся система может находиться в работоспособном состоянии после отказа основного устройства только при условии безотказной работы остальных блоков. При расчетах общей интенсивности отказов системы, равной сумме интенсивностей отказов составляющих ее модулей, принимается , что $\lambda_{ук}(t)$, $\lambda_{ру}(t) << \lambda_{оу}(t)$.

Резервирование с постоянно включенным («горячим») резервом предполагает, что все модули системы начинают функционировать одновременно, начиная с момента включения системы. Такой способ наиболее часто реализуется в системах на основе мажоритарного определения выходного сигнала (Y_k): по способу «большинства». Пример реализации показан на рисунке 1.3.

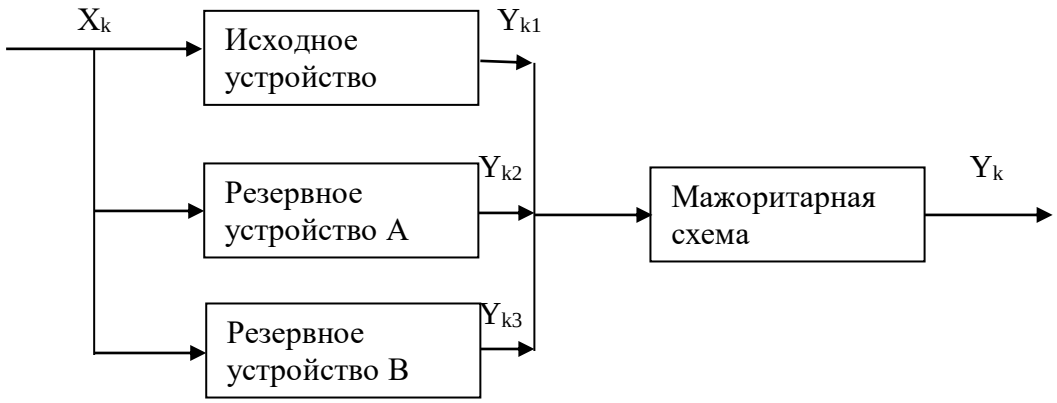


Рис.1.3. Обобщенная структурная схем с резервированием на основе мажоритарного способа определения выходного сигнала

При таком способе резервирования предполагается, что резервные устройства (модули) полностью идентичны исходному устройству, и на их выходах при безотказном функционировании должны вырабатываться одинаковые сигналы (последовательности). Выходной сигнал резервированной системы (Y_k) формируется по принципу большинства: один из трех (при одинаковых сигналах Y_{k1}, Y_{k2}, Y_{k3}) или один из двух (если сигнал на выходе одного из трех устройств отличается от двух других). Предполагается, что интенсивности отказов в исходном и в резервных устройствах одинаковы: $\lambda_{оу}(t) = \lambda_{руА}(t) = \lambda_{руВ}(t)$ и $\lambda_{оу}(t) >> \lambda_{мс}(t)$, где $\lambda_{мс}(t)$ – интенсивность отказов мажоритарной схемы. Исходя из этого (интенсивностью отказов мажоритарного устройства пренебрегаем), можно рассчитать вероятность безотказной работы системы:

$P(t) = P_3(t) + P_2(t)$,

где $P_3(t)$ – вероятность того, что все три устройства (исходное и два резервных) работают безотказно, $P_2(t)$ – вероятность того, что два устройства из трех указанных работают безотказно.

Нетрудно подсчитать, что

$P_3(t) = P_{оу}(t) * P_{руА}(t) * P_{руВ}(t)$,

$P_2(t) = P_{оу}(t) * P_{руА}(t) * (1 - P_{руВ}(t)) + P_{оу}(t) * P_{руВ}(t) * (1 - P_{руА}(t)) + P_{руА}(t) * P_{руВ}(t) * (1 - P_{оу}(t)) = 3 (P_{оу}(t))^2 * (1- P_{оу}(t))$.

Из сделанного анализа принципов функционирования резервированных устройств, видно, что они характеризуются также временной избыточностью: формирование выходного сигнала требует дополнительного времени. Однако классическими системами с временной избыточностью принято считать такие,

которые характеризуются также информационной избыточностью на основе использования специальных кодов, обнаруживающих и корректирующих ошибки в информационных последовательностях (словах). Об этом более подробно речь пойдет в разделе .

1.4. Методы и технологии обеспечения надежности программных комплексов

В составе «мягкого оборудования» (software) аппаратно-программных комплексов (АПК) можно выделить три части: *математическое обеспечение* (МО), *информационное обеспечение* (ИО) и *программное обеспечение* (ПО). Первую часть (МО) составляет совокупность математических моделей, методов и алгоритмов, на основе которых АПК выполняет задачи в соответствии с целевым назначением. ИО есть совокупность баз данных, файловых структур и других элементов, определяющих формирование, подготовку и условия обработки данных. И, наконец, программное обеспечение АПК реализует его МО на основе ИО. Следует также подчеркнуть, что методы преобразования информации могут быть реализовано как аппаратно, так и программно либо аппаратно-программно.

Последствия ошибок в «мягком оборудовании» не отличаются от последствий отказов аппаратных средств. В этом смысле говорят об отказах и надежности программных средств (а равно – и о надежности МО, ИО).

Обеспечение корректности программ и данных означает недопущение ошибок в элементах МО, ИО, ПО до начала их эксплуатации. Компьютерная программа как результат реализации алгоритмов на языке программирования считается корректной, если она удовлетворяет своим функциональным спецификациям.

Основными причинами ошибок в ИО являются необнаруженные при подготовке данных и их вводе неточности, ошибки адресации, ошибочные константы, операнды и т.д., а также искажения данных вследствие несанкционированного доступа к системе.

К основным причинам ошибок ПО относятся следующие: 1) незамкнутость реализованного алгоритма, вызванная ошибками МО, 2) синтаксические ошибки, допущенные при реализации алгоритма на конкретном языке программирования, неправильное применение конструкций языка, системные ошибки, 3) неправильная отладка программы после выявления ошибок, которая приводит к новым ошибкам.

Методы обеспечения безошибочности данных могут реализовываться при проектировании программных комплексов. К таким методам относят применение автоматических процедур проверки корректности данных (например, проверку совместимости элементов структур данных); дублирование работ при подготовке данных и др.

Методы обеспечения безошибочности программ являются одной из важнейших характеристик технологий программирования. К основным из таких методов относят снижение сложности программ, широкое применение систем

автоматизации проектирования ПО, применение языков программирования высокого уровня и др.

Основным средством уменьшения числа ошибок, допущенных разработке ПО, является отладка. При создании условий для наиболее эффективного выявления ошибок следует иметь в виду, что ПО не отказывает, когда не находится в работе.

П определению программа считается отказоустойчивой, если она обеспечивает полезную работоспособность не ниже заданного минимального уровня при наличии программных ошибок. Разработка ПО, нечувствительного к определенным типам ошибок и искажений, считается более перспективным направлением по сравнению с разработкой безошибочных программ.

1.5. Общая характеристика безопасности информационных систем

Безопасность системы/процесса характеризует способность системы противодействовать несанкционированному доступу к нему вне зависимости от целей этого доступа. Параметрами безопасности являются: *конфиденциальность*, *целостность*, *доступность*.

Конфиденциальность информации – свойство информации быть известной только допущенным и прошедшим авторизацию субъектам системы (пользователям, программам, процессам и др.).

Целостность – состояние данных или компьютерной системы, в которой данные и программы используются установленным способом, обеспечивающим устойчивую работу системы и единство данных.

Доступность компонента (ресурса) системы – свойство компонента (ресурса) быть доступным для использования авторизованными субъектами системы в любое время в соответствии с установленным регламентом.

Принято считать, что *информация стала самым дорогим продукт*ом в сфере межличностных отношений. Именно поэтому информация и информационные системы все чаще становятся объектами атак (несанкционированного доступа).

Различают следующие основные методы несанкционированного доступа: 1) физические (например, простое копирование), 2) программные (вирусы, «троянские кони», клавиатурные анализаторы, анализаторы протоколов и другие деструктивные программы), 3) электронные (на основе исследования электромагнитного излучения от различных блоков и устройств компьютерной техники, а также каналов передачи информации), 4) технические (непосредственное подключение к каналам связи).

Все многообразие методов и средств противодействия несанкционированному доступу условно подразделяется на следующие группы.

1. **Организационные методы и средства** подразумевают разработку, и исполнение в любой организации/лаборатории правил, регламентирующих и регулирующих доступ физических лиц к информации, хранящейся на носителях либо передаваемой внутри сети данного предприятия. В организациях, осуществляющих операции над критической информацией (правительственные,

государственные, банковские, коммерческие и иные структуры) назначается специальное ответственное лицо — администратор безопасности (АБ), ответственный за реализацию и соблюдение правил на основе реализуемой политики безопасности.

2. **Правовые методы**. Гражданский правовой кодекс предусматривает наказание за *компьютерные преступления*. В 1983г. Организация экономического сотрудничества и развития определила под термином «компьютерная преступность» (или «связанная с компьютерами преступность») любые незаконные, неэтичные или неправомерные действия, связанные с автоматической обработкой или передачей информации. Практически во всех странах с развитой информационной инфраструктурой (в том числе и в Беларуси) предусматривается уголовно-правовая защита от компьютерных преступлений.

3. **Физические методы.** Объединяют методы ограничения физического доступа лиц к каналам передачи информации, устройствам ее хранения и обработки. Основаны на использовании простых замков, магнитных карт, чипов, таблеток, анализе антропометрических и биологических параметров человека (сетчатка глаза, отпечатки пальцев и др.).

4. **Программно-технические методы.** Основаны на использовании аппаратных и/или программных средств, позволяющих идентифицировать пользователя (либо техническое средство), а также оценить происхождение программного средства, поступающего в информационную сеть. Наиболее известными из указанных средств являются: использование пароля, антивирусных программ, брандмауэров или «огненных стен» ((firewalls) на входе сети, криптографического преобразования информации на основе методов шифрования.

Далее в пособии в основном будем анализировать методы именно этой группы. Однако здесь подчеркнем простой и неоспоримый факт: абсолютно защищенные персональный компьютер или компьютерная сеть, операционная система, прикладная программа – такая же иллюзия, как и абсолютно надежно охраняемый дом. Защита ваших данных и каналов связи с Интернетом от воздействия *интрузов* (нежелательных программ, или физических лиц) является, по определению, результатом известного компромисса. Этот компромисс базируется на важнейшем и универсальном подходе к разработке и реализации политики защиты: защита информации только тогда является оправданной и разумной, когда стоимость реализации политики безопасности, по крайней мере, не меньше стоимости потерь, вызванных несанцкционированным ее использованием.

Организация Microsoft Security Response Center сформулировала наиболее важные положения по компьютерной безопасности, которые выглядят следующим образом (приводятся здесь без комментариев):

1. Если злоумышленник убедит вас в том, что его программа должна выполняться на вашем компьютере, этот компьютер перестанет принадлежать вам.
2. Если злоумышленник сможет изменить операционную систему на вашем компьютере, этот компьютер перестанет вам принадлежать.

3. Если хакер (или кракер) имеет физический доступ к вашему компьютеру, этот компьютер перестанет вам принадлежать.
4. Если злоумышленник загрузит свои программы на ваш web-сайт, этот сайт перестанет вам принадлежать.
5. Использование простых паролей ослабляют строгие меры безопасности.
6. Компьютер безопасен только в том случае, когда администратор (безопасности) добросовестно относится к своим обязанностям.
7. Шифрование данных является безопасным в той мере, в какой соблюдены правила безопасного хранения ключа.
8. Устаревший сканер вирусов лишь немногим лучше отсутствующего сканера вирусов.
9. Абсолютная анонимность лишена практического смысла как в реальной жизни, так и при работе в Интернете.
10. Любая технология не является панацеей от всех бед.

Вопросы для самоконтроля

1. Дайте определения основных понятий и терминов из области информационной безопасности и надежности систем.
2. Приведите соотношения для расчета вероятности безотказной работы ИВС, вероятности отказа ИВС и интенсивности отказов в ИВС.
3. Рассчитайте вероятность безотказной работы и вероятность отказа системы, структура которой приведена на рисунке 1.2 (при заданных преподавателем интенсивностях отказов в каждом из модулей).
4. Рассчитайте вероятность безотказной работы и вероятность отказа системы, структура которой приведена на рисунке 1.3 (при заданных преподавателем интенсивностях отказов в каждом из модулей).
5. Охарактеризуйте надежность программного обеспечения.
6. Проанализируйте угрозы для ИВС
7. Охарактеризуйте методы защиты ИВС от несанкционированного доступа.

Задание для самостоятельного выполнения

Создайте программное средство, с помощью которого произведите сравнительную оценку надежности ИВС на рисунках 1.2 и 1.3 при заданных преподавателем условиях.

2. ИНФОРМАЦИЯ, ЕЕ КОЛИЧЕСТВО И КАНАЛЫ ПЕРЕДАЧИ

2.1. Характеристики системы передачи информации

Передача информации (данных) осуществляется между двумя абонентами, называемыми *источником сообщения* (ИС) и *получателем сообщения* (ПС). Источником и получателем могут быть люди либо технические средства. ИС и ПС обмениваются информацией посредством канала передачи. Таким образом, простейшая информационная система состоит из трех перечисленных элементов. Ее обобщенная структурная схема приведена на рисунке 2.1.

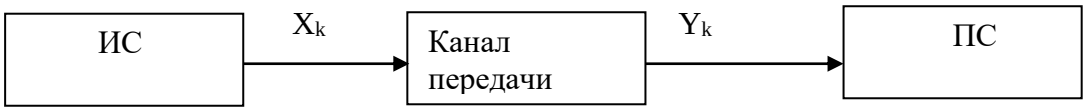


Рис.2.1. Обобщенная структурная схема информационной системы (системы передачи информации)

ИС и ПС обмениваются информацией в технических системах в виде сигналов, сформированных на основе определенного *алфавита*. Характеристикой алфавита является его *мощность,* N_a — количество символов, на основе которых формируется сообщение. Например, мощность английского алфавита — 26 символов, русского — 33 символа, мощность алфавита, на основе которого функционируют и взаимодействуют между собой компьютеры, составляет 2 символа (0 и 1).

Другой важной характеристикой источника (и получателя) сообщений является тип используемого сигнала. Существуют 2 основных типа сигналов: 1) аналоговый, 2) цифровой. Аналоговый сигнал, в отличие от цифрового, характеризуется бесконечным числом состояний (значений). ИС на основе аналогового сигнала называется *источником непрерывных сообщений*. ИС на основе цифрового сигнала называется *источником дискретных сообщений*. Примером последнего может служить источник двоичных (бинарных) сообщений.

Далее будем рассматривать только ИС и ПС дискретных сообщений. Наибольший интерес представляет собой цифровой сигнал на основе алфавита A{0,1}.

В произвольном сообщении символы алфавита могут появляться с различной вероятностью. Если длина сообщения достаточно велика, то статистический анализ этого сообщения позволит получить вероятностные характеристики этого алфавита. Очевидно, что различные символы в произвольном сообщении (особенно при $N_a>2$) появляются с различной вероятностью, т.е. существуют символы с минимальной и максимальной вероятностью появления. Например, подсчитано, что наиболее часто (в 13% случаев) в документах на английском языке ($N_a = 26$) появляется буква «e», а наиболее редко (в 0,7% случаев) – буквы «x», «y» и «z». Можем записать, что

вероятность того, что произвольный символ ξ произвольного документа (текст, база данных, текст программы) будет буквой «е» (или другой из указанных букв) можем записать как

$$\mathrm{P}(\xi = e) = 0.13,$$

$$\mathrm{P}(\xi = x) = \mathrm{P}(\xi = y) = \mathrm{P}(\xi = z) = 0.007.$$

Информационной характеристикой алфавита (источника сообщений на основе этого алфавита) является *энтропия*. Этот термин применительно к техническим системам был введен Шенноном и Хартли.

Энтропию алфавита A{a_i} по Шеннону рассчитывают по следующей формуле:

$$\mathrm{H}_S(A) = -\sum_{i=1}^{N} P(a_i) * \log_2 P(a_i), \qquad (2.1)$$

где $i = \overline{1, N}$, a_i – элемент алфавита, $P(a_i)$ – вероятность $P(\xi = a_i)$.

Заметим, что $\sum_{i=1}^{N} P(a_i) = 1$.

С физической точки зрения *энтропия показывает, какое количество информации приходится в среднем на один символ алфавита*.

Частным случаем энтропии Шеннона является энтропия Хартли. Дополнительным условием при этом является то, что все вероятности одинаковы и постоянны для всех символов алфавита. С учетом этого формулу (2.1) можно преобразовать к виду:

$$H_{Ch}(A) = \log_2 N.$$

Например, энтропия Хартли для латинского (английского) алфавита составляет 4,7 бит.

Если подсчитать энтропию Шеннона и энтропию Хартли для одного и того же алфавита, то они окажутся не равными. Это несовпадение указывает на избыточность любого алфавита (при N>2).

Сообщение M, которое состоит из n символов, должно характеризоваться определенным *количеством информации*, I(M):

$$\mathrm{I(M)} = \mathrm{H(A)} * \mathrm{n}. \qquad (2.2)$$

Нетрудно предположить и просто убедиться, что количество информации в сообщении, подсчитанное по Шеннону, не равно количеству информации, подсчитанному по Хартли. На основе этого парадокса строятся и функционируют все современные системы сжатия (компрессии) информации.

Пример фрагмента кода программы, позволяющей вычислить энтропию английского алфавита, приведен ниже (листинг 1.1).

```
                        //подключение необходимых для //работы
                        программы библиотек
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
```

```
#include <ctype.h>
int main()        // тело главной функции
{char ch;                //переменная, которая будет
                         //хранить считываемый из файла
                         //символ
int count[26];    //массив coun[]t, элементы
                         //которого будут содержать
                         //количество появлений в
                         //информационном сообщении
                         //некоторого символа
double p[26];            //массив вероятностей появления
                         //символов алфавита
int t;                   //переменная, которая будет
                         //хранить общее количество
                         //символов файла (текстового
                         //сообщения), включая символ
                         //конца строки
t=0;
char alpha[]={'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
'y', 'z'};
for (int i=0;i<26;i++)
  {count[i]=0;}
                         //инициализация элементов
                         //массива count[] нулями
ifstream in ("b");       //открытие файла для чтения
if (!in) {
cout<<"Cannot \n";return 1;
                         //проверка возможности открытия
                         //файла
}
  while(in){       //проверка достижения
  //конца файла
  in.get(ch);            //считывание очередного символа //из
                         файла
  t++;                   //увеличение счетчика общего //количества
                         символов файла //(включая пробел) на 1
for (int i=0; i<26; i++)
  {                      //приведение всех символов //файла к
                         нижнему регистру и //проверка вновь
                         считанного //символа на соответствие
                         //символам исходного алфавита
if (putchar(tolower(ch))==alpha[i])
                         //увеличение счетчика появления //символа
                         на 1
count[i]=count[i] + 1;
for (int i=0;i<25;i++)
{
double p[i]=count[i]/(t - 1);
                         //массив     p[]     содержит     значения
                         //появления символов алфавита
                         //установки на вывод числа с //точкой
cout.setf(ios::showpoint);
cout.setf(ios::fixed);
cout.precision(2);
```

```
cout<<alpha[i]<<"* - *";
cout<<count[i]<<"* - *"<<p[i]<<"\n";
}
double h;
h=0;
for (int i=0;i<26;i++)
{double result;
if (p[i]!=0)            //если вероятность появления //символа не
                        равна 0 (условие //для вычисления
                        логарифма)
{result=log(p[i]);
                        //вычисление энтропии Шеннона
                        //(переменная h)
h=h + (p[i]*result);
}
}
  h=-h;                 //взятие энтропии с //противоположным
                        знаком
                        //и вывод энтропии
cout<<"h="<<h<<"\n";
int V=0;
  V=h*(t - 1);          //вычисление количества //информации
cout<<"I="<<V<<"\n";
                        //вывод значения количества //информации
in.close();             //закрытие файла
return 0;
```

Листинг 1.1. Пример фрагмента кода программы в C++, позволяющей вычислить энтропию английского алфавита

2.2. Двоичный канал передачи информации

Как было подчеркнуто выше, двоичный канал передачи информации является дискретным – он основан на алфавите, состоящем из двух символов: 0 и 1 - A{0,1}. Используя (2.1), вычислим энтропию этого алфавита:

$H(A_2) = -P(0)*\log_2(P(0)) -P(1)*\log_2(P(1)$. (2.3)

К примеру, полагая что сообщение M состоит только из единиц (M=11…1) и имеет длину n: $M = \underbrace{11\ldots11}_{n}$, т.е. вероятность того, что произвольный символ равен единице, составляет единицу (P(1)=1), и другая вероятность - P(0)=0 для $i = \overline{1,N}$. Фактически, здесь имеет место использование моноалфавита: алфавита, состоящего из одного символа.

Если в этом случае подставить в (2.3) соответствующие значения, то получим, что энтропия моноалфавита равна 0 бит, количество информации в сообщении из единиц (либо из нулей) также составляет 0 бит. Этот практический результат поясняет физический смысл понятия информации в теории Шеннона: *информацией является лишь такое сообщение, которое снимает некоторую неопределенность, т.е содержит новые для получателя данные*. Если априори

известно, что сообщение будет состоять из набора одинаковых символов, то для получателя сообщения оно никакой неопределенности не содержит.

Если для бинарного алфавита вероятность появления в произвольном сообщении одного их этих символов стремится к нулю (или равна ему), то энтропия такого алфавита также будет стремится к (или равняться) нулю.

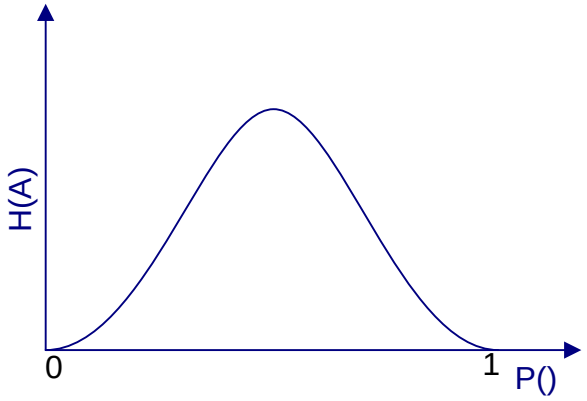


Рис.2.2. Качественная характеристика энтропии бинарного алфавита при различных значениях вероятностей появления символов в сообщении

Между этими точками значение функции (энтропии) должно пройти через максимум (т.к. энтропия не может быть отрицательной). Для нахождения этого максимального значения надо найти производную:

$$\frac{\partial H(A)}{\partial P(1) * \partial P(0)} \Rightarrow Max\ H(A_2) = \begin{cases} P(0) = \frac{1}{2} \\ P(1) = \frac{1}{2} \end{cases},$$

$$Max\ H(A_2) = -\frac{1}{2} * \log_2 \frac{1}{2} - \frac{1}{2} * \log_2 \frac{1}{2} = 1\ бит.$$

Таким образом, энтропия бинарного алфавита принимает максимальное значение, равное 1 бит, при условии равновероятного появления каждого символа алфавита в сообщении (см. рисунок 2.2).

Интересным представляется оценка количества информации в сообщении, выполненная на основе различных подходов.

Пример2-1. Анализируем сообщение M = ‘We are happy’ (пробелы не учитываем). Если принять, что энтропия английского алфавита, вычисленная по Шеннону, составляет 4,2 бит (примерно), то в анализируемом сообщении содержится 42 бита информации: I(M)=4.2*10=42 бита.

С другой стороны, предполагая, что сообщение переведено в ASCII коды (один символ алфавита заменяется соответствующим байтом двоичных символов) и, предположив, что P(0) = P(1) = 0,5, получаем $I(M_{ASCII})$=1*80=80 бит.

Данный пример является свидетельством и подтверждением избыточности не только алфавита, но и сообщений, сформированных и обрабатываемых в компьютерных системах, т.е. любые сообщения характеризуются

информационной избыточностью, что позволяет сжимать их без потери информации.

Вопросы для самоконтроля

1. Дайте определения мощности и энтропии алфавита.
2. Как рассчитываются энтропия Шеннона и энтропия Хартли? В чем принципиальное различие между этими характеристиками? Дайте толкование физического смысла энтропии.
3. Изобразите обобщенную структурную схему системы передачи информации. Поясните назначение и особенности элементов системы.
4. Что такое избыточность алфавита и избыточность сообщений, сформированных в компьютерных системах? Принцип действия каких систем основан на существовании данной избыточности?
5. Расположите в порядке возрастания энтропии известные вам алфавиты.

Задание для самостоятельного выполнения

Создайте программное средство, с помощью которого произведите вычисление энтропии заданного алфавита и вычислите количество информации, содержащейся в ваших фамилии, имени и отчестве.

Примечание.

Файл выходных данных (result.txt) должен содержать таблицу вероятностей появления символов алфавита, значения энтропии заданного и двоичного алфавитов, количество информации для исходного и преобразованного файлов.

Символами русского алфавита считать символы а(А)…я(Я), английского – a(A)…z(Z).

Все символы алфавита должны быть приведены к одному регистру (прописные/строчные). Следует отметить, что символами алфавита не являются служебные символы и цифры, но при вычислении количества информации в сообщении эти символы следует включить в общее количество символов сообщения. Программа может быть построена на основе листинга 1.1.

Для перевода информационного сообщения в двоичный код можно использовать код ASCII каждого символа, к которому следует применить функцию *itoa*().

Функция *itoa*() конвертирует целое число *num* в строчный эквивалент и помещает результат в строку, на которую указывает параметр *str*. Основание системы счисления для записи выходной строки определено параметром *radix,* который может принимать значения в интервале от 2 до 36. Функция *itoa*() возвращает указатель на *str*. Прототип функции *itoa*() содержится в файле *stdlib.h.*

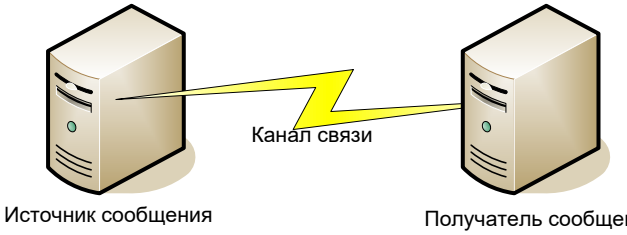
Синтаксис функции: *char *itoa(int num, char *str, int radix).*

Например, результат работы функции itoa(num,buf,2) будет сохранен в переменной buf=”11”.

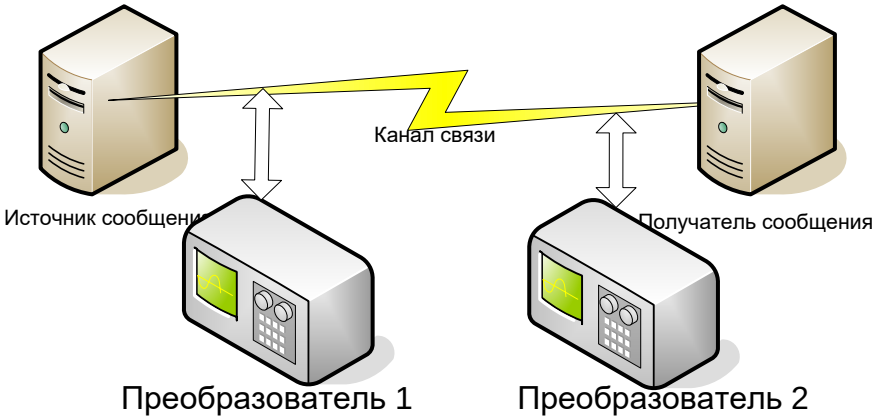
Базовые методы преобразования информации

Цели и задачи преобразования информации.

Цель любого преобразования — улучшение параметров, характеризующих качество хранения, передачи и обработки информации либо данных. Параметры качества всегда связаны с другими параметрами.



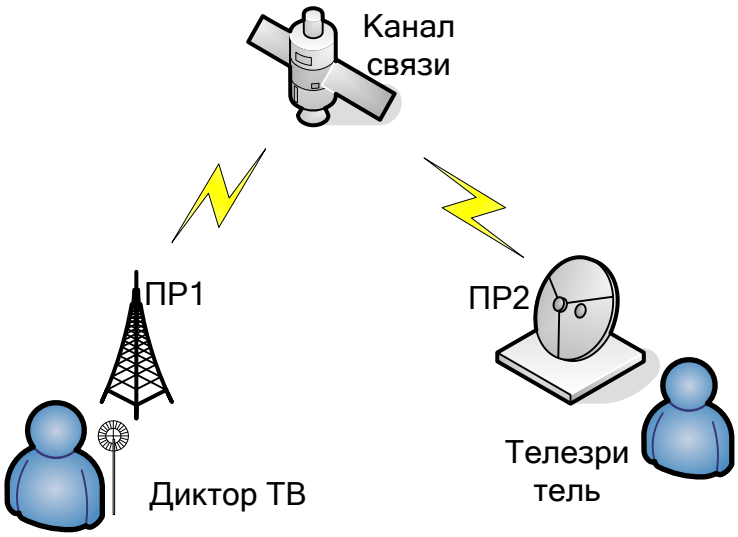
Получатель и источник сообщения не обязательно могут быть однотипными, поэтому преобразование информации используется практически всегда. В наиболее простом случае такой канал дополняется двумя блоками:



Преобразователь 1 (ПР1) выполняет функции адаптации исходного сообщения под исходный канал. Преобразователь 2 (ПР2) — наоборот.

Также ПР1 может выполнять несколько функций: кодирование источника и кодирование канала; аналогично и ПР2.

К примеру:



В современных информационных системах качество передачи информации достигается с помощью трех базовых методов:

1. Методы помехоустойчивого кодирования
2. Методы сжатия (компрессии) данных
3. Криптографические преобразования

Все эти методы могут быть использованы независимо и автономно, либо в произвольной комбинации, в зависимости от того, какая преследуется цель.

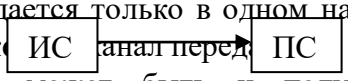
Методы помехоустойчивого кодирования обеспечивают повышение надежности хранения или передачи информации в случае преднамеренных (не преднамеренных) помех. То есть достигается повышение качества передачи информации (или надежность хранения) путем обнаружения ошибок.

Методы сжатия предназначены для физического снижения объема информации для ее хранения или передачи. Улучшение параметров достигается за счет снижения времени передачи информации по каналам фиксированной скорости и некоторое повышение конфиденциальности информации.

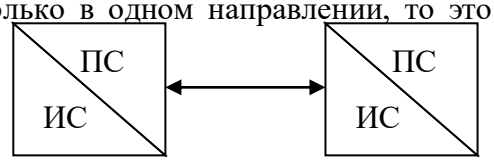
Криптографические методы преобразования направлены на повышение уровня конфиденциальности информации. Это одни из методов ограничения доступа к информации.

Помехоустойчивое кодирование информации

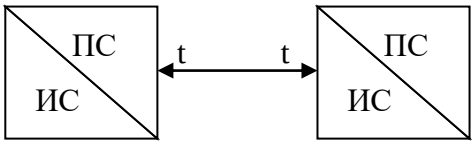
Особенности современных каналов передачи данных:

- Если сигнал передается только в одном направлении, то это односторонний, **симплек**


- Если устройство может быть и получателем и источником сообщения, и информация может передаваться в обоих направлениях. Но в фиксированный момент времени информация передается только в одном направлении, то это **полудуплексный** канал.



Если ИС и ПС могут обмениваться сообщениями в любой момент времени и получать ее в это же время — это **дуплексный** канал.



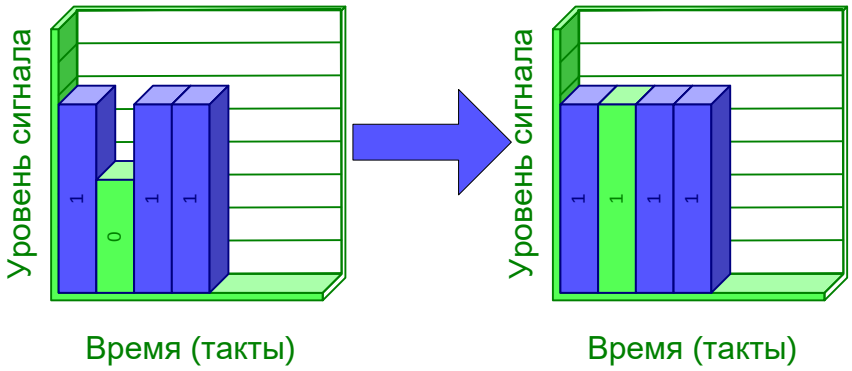
Характеристика надежности передачи данных по каналам симплексного, полудуплексного и дуплексного типов.

Рассмотрим системы передачи сигнала на основе бинарных каналов. В различных системах вид сигнала может быть разным и определяется качеством сигнала.

Качество (надежность) **канала** можно определить вероятностью появления ошибочных символов на входе получателя информации. Т.е. вероятностью того, что получено сообщение Y`, а отправлено X.

К примеру, пусть сообщение X = 1011 и в канале действуют непреднамеренные (преднамеренный) «белый шум» и «шум Гаусса», которые характеризуются очень широким спектром и мощность и также практически одинаковы во всем спектре.

На входе принятый сигнал трактуется как Y` = 1111.Второй символ принят с ошибкой, что говорит, что произошла одиночная ошибка.



Ошибка — несовпадение передаваемых и получаемых символов в одинаковой позиции. В рассмотренном примере ошибка произошла в 3-м бите (рассматривая последовательность справа налево). Возможна ситуация когда помеха и сигнал взаимно компенсируются, т.е. в какой-то момент времени помеха и сигнал имели взаимно обратные фазы и примерно равные амплитуды. Помехи могут быть вызваны различными причинами и иметь различную структуру, но их

негативное влияние от этого не меняется. Практически все современные информационные системы вне зависимости от организации методов обмена сообщениями имеют средства обнаружения и коррекции возникающих ошибок.

Методы нейтрализации ошибок можно разделить на два больших класса:

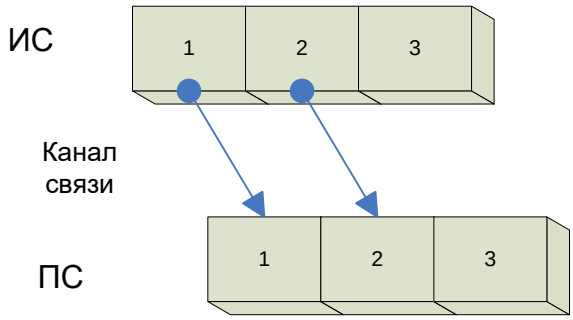
- На основе временной избыточности;
- На основе аппаратной избыточности.

В критических системах возможна организация обнаружения ошибок на основе использования двух этих методов.

Временная избыточность

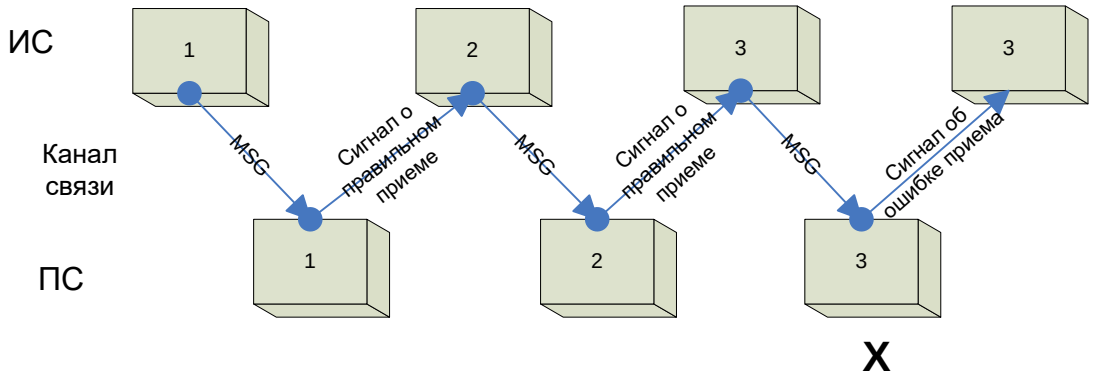
Организация обмена в общем случае реализована по каналам связи (рисунок выше). Обычно источник сообщения передает информацию порциями — кадрами, пакетами и т.д. Поэтому можно выделить несколько методов коррекции ошибок.

Запрос с остановкой. Предположим что задержка сигнала по времени между источником и получателем равна времени передачи одного пакета.



На рисунке изображен общий случай передачи, когда нет никаких дополнительных запросов.

Метод повтора запросов характеризуется тем, что источник начинает передачу очередного пакета только после получения информации о качестве предыдущего принятого пакета. В зависимости от этого сигнала будет передан очередной пакет или повторен предыдущий. Таким образом, общая схема передачи трансформируется к следующему виду:

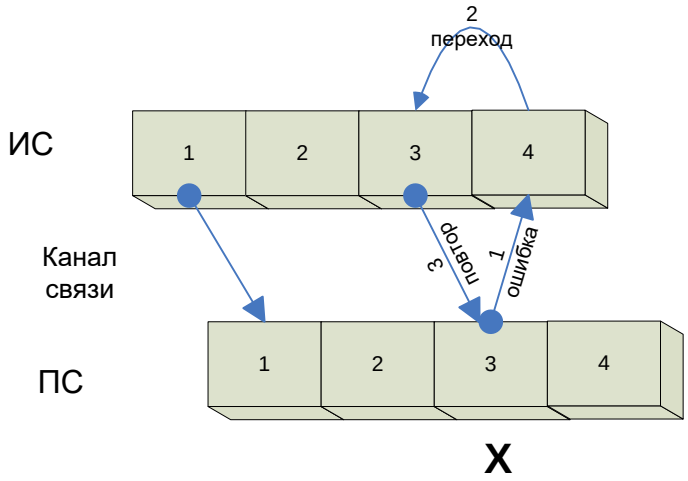


По такому принципу работают сети на основе Token Ring.

Если получатель сообщения не должен и не может прочитать сообщение (сжато, зашифровано) то используются средства CRC32.

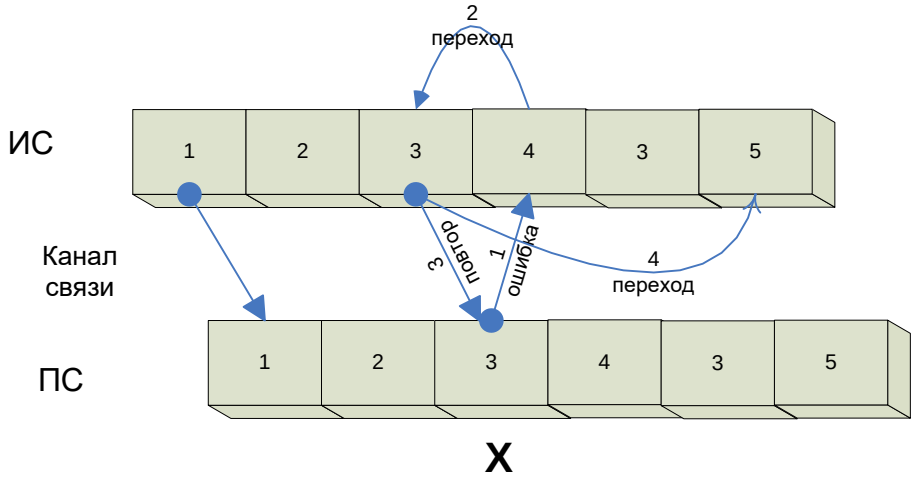
Основные особенности такого алгоритма — максимальное качество конечного фрагмента и минимальные скорость и пропускная способность.

Непрерывный запрос с возвратом. Суть метода в том, что источник передает информацию непрерывно, если же он при этом получает сообщение об ошибочном приеме ранее переданного, то возвращается обратно к ошибочному сегменту и передача начинается с него.



Если первый метод можно реализовать на основе симплексных каналов, то этот метод требует применения как минимум полудуплексных каналов передачи, что повышает скорость передачи данных.

Непрерывный запрос с выборочным повтором. В отличие от предыдущего метода в этом передаются только блоки данных с ошибкой, без повтора последующих.



По такому принципу основам обмен данными в сети интернет. Передающая сторона повторяет только пакет, который запросил тот, кто принимает информацию.

Все этим методы требуют дополнительных временных затрат. Суммируя все что было сказано выше:

1. Метод запрос с остановкой является наименее быстродействующим и используется, как правило, для передачи информации не в «реальном времени».
2. Наиболее скоростным из рассмотренных методов является третий, непрерывный запрос с выборочным повтором, но он требует как минимум полудуплексный канал передачи. На основе этого метода работает Internet и протокол TCP.

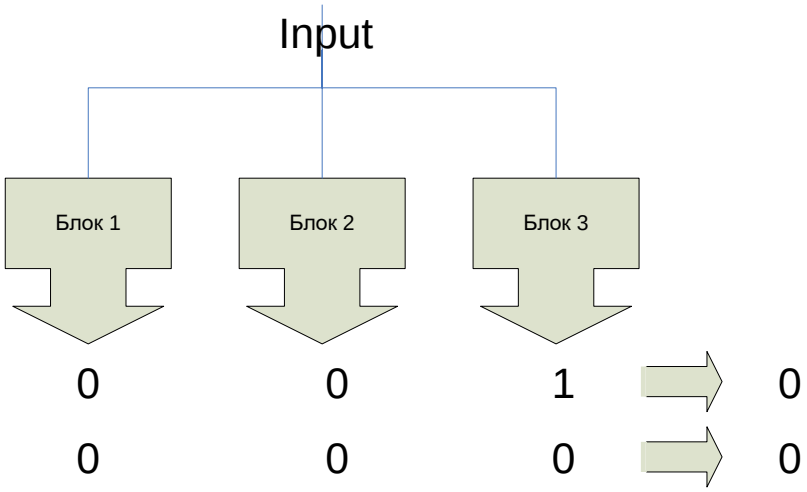
Структурная избыточность

Наиболее простым методом со структурной избыточностью является метод **простого резервирования**. Суть метода в полном дублировании целых блоков или систем. Организация обработки и передачи информации в таких случаях может быть реализована следующим образом.

«Холодный» резерв. Количество резервных модулей может быть различным, от 1 до N (обычно N=3) «Холодный» означает, что в определенный момент времени t_i функционирует только один модуль/блок. Если в момент времени $t_i+\Delta t$ штатный режим функционирования основного модулю системы нарушается, то его функции начинают выполнять резервные блоки. Этот принцип реализуется, как правило, в бортовых системах (космические, авиационные системы), в системах управления атомными станциями и т.п. Один блок работает, другие законсервированы.

Принцип «холодного» резерва можно расширить от модулей и блоков до систем.

«Горячий» резерв. Характеризуется тем, что параллельно и не зависимо друг от друга один и тот же алгоритм реализуется на нескольких (N) блоках или системах. Окончательный вариант выбирается по принципу «голосования». Как правило, в таких системах используют нечетное количество дублирований блоков системы (3,5,7,…). Обычно у этих блоков один вход, т.е. одни и те же данные поступают на вход каждого блока. К примеру, для 3-х блочной системы с бинарным выходом:



Структурно временная избыточность

Эти методы основаны на использовании специальных методов обработки информации — **методов помехоустойчивого кодирования информации**.

Суть метода в преобразовании исходного информационного сообщения X_k k — длина двоичных символов сообщения. К X_k дополнительно присоединяют избыточные символы длиной r бит. Таким образом, формируют кодовое слово X_n длина этого слова n=k+r двоичных символов. Кодовое слово X_n состоит из информационного X_k, и дополнительной избыточности X_r. Информацию содержит только информационное слово, избыточность не несет никакой полезной информации для получателя сообщения. Назначение избыточности X_r — обнаружение и исправление ошибок.

В зависимости от принципа вычисления дополнительных символов и их числа реализуются различные алгоритмы помехоустойчивого кодирования. Таким образом, избыточность X_r генерируется на передающей стороне и используется принимающей для обнаружения и/или исправления ошибки передачи информации.

Основные понятия теории помехоустойчивого кодирования информации

Ошибка — несоответствие символа принятого и переданного сообщения.

Вес по Хеммингу {W(X=1101) = 3} — количество ненулевых символов в слове X.

Расстояние по Хеммингу {D(X,Y)} — количество позиций, в которых X и Y отличаются между собой. Как правило, длина сравниваемых слов должна быть одинакова. Пример: D(X=101,Y=111) = 1

Длина слова — это количество бинарных символов, из которых состоит слово.

Расстояние по Хеммингу можно вычислить как вес от суммы по модулю 2 этих двух слов. D(X,Y) = W(X⊕Y)

Пример 1: X=1011 Y=0000

D(X,Y)=3

$$\begin{array}{r} 1011 \\ 0000 \\ \hline W(1011) = 3 \end{array}$$

Пример 2: X=1111 Y=1111

D(X,Y)=0

W=0

Длина слова и расстояние Хемминга — основополагающие понятия в теории помехоустойчивого кодирования информации.

Простейший избыточный код (ПИК)

Простейший избыточный код является простейшим контролем сообщения на четность нечетность. Количество избыточных символов всегда равно 1 и не зависит от значения k.

Значение этого символа будет нулевым, если сумма всех символов кодового слова по модулю 2 равна нулю. Тогда говорят о просто четности.

Пример: X_k=10101 $\quad X_r = \sum_i^N X_i = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$

То есть слово X_N составленное из X_k и X_r будет равно 101011

$W(X_N)$= четное

Значение X_r в данном алгоритме — обнаружение ошибки. Код простой четности/нечетности позволяет обнаруживать все нечетные ошибки (нечетное число ошибок), но не позволяет их исправлять.

В общем случае код позволяет обнаруживать t_0 ошибок, где $t_0 = \frac{d}{2}$ если d — четно и $t_0 = \frac{d-1}{2}$, если d — нечетно. Обобщив сказанное можно записать

$$t_i = \begin{cases} \frac{d-1}{2} & d \to нечетное \\ \frac{d-2}{2} & d \to четное \end{cases}$$

Рассмотри расстояние между двумя произвольными словами в коде простой четности. Пусть k=4

X_k	X_r	
0000	0	$\min d = 2$
0001	1	$d = 2$
0010	1	$d = 3$
⋮	⋮	
0111	1	$d = 4 \rightarrow Max$
1111	0	

В соотношениях приведенных ниже значение d — минимальное кодовое расстояние между двумя произвольными кодовыми словами, принадлежащими коду. В рассмотренном примере, где k=4, всего имелось 2k кодовых слов принадлежащих коду.

Из тех же соотношений видно, что данный код позволяет обнаруживать любое нечетное количество ошибок, но не позволяет исправлять их.

Пример: X_k=1101; X_r=1; X_N=11011

Это слово X_N будет передаваться по каналам связи от источника сообщений получателю сообщения. Пусть Y_N=10011 — это полученное слово с ошибкой во втором разряде. Задача системы на стороне приемника сообщений — проанализировать полученное сообщение и выявить ошибку. Источник и получатель сообщения должный работать по одинаковому алгоритму, т.е. использовать один и тот же код для дополнения и анализа сообщений.

Получатель вычисляет дополнительное слово, которое является сверткой по модулю 2: $Y'_N = \sum Y_{N_i} = 1 \oplus 0 \oplus 0 \oplus 1 = 0$

Если полученный символ соответствует дополнительному символу — контроль ошибок пройден, ошибок нет, иначе — есть ошибка и необходима повторная передача сообщения.

Сразу видно, что при наличии нечетного (1,3,5…) количества ошибок тест даст правильный результат, иначе же, при четном количестве ошибок, тест не покажет наличие ошибок.

Любой избыточный код по сути характеризуется тремя основными показателями:

- n — длина слова;
- k — длина информационного слова;
- r — длина проверочного (дополнительного) слова.

На основе этих параметров можно записать:

n=k+r

R=k/n — относительная скорость передачи информации с использованием кода.

Чем больше n, тем меньше R. **Ru=r/k** — **избыточность** кода.

В соответствии с этим соотношением можно сформулировать основную проблему кодирования: необходимо найти коды с максимальной возможной скоростью передачи информации при минимальной избыточности кода. Это требование противоречиво, т.е. перед нами стоит задача оптимизации.

С практической точки зрения использование кода подразумевает реализацию определенных алгоритмов кодирования и декодирования сообщений. Кодирование означает, что кодовое слово X_k должно быть преобразовано к X_n на основе существующего кода за определенное время. Декодирование означает, что принятое Y_n надо преобразовать к X_k, учитывая ошибки передачи и особенности алгоритма.

В общем случае, с математической точки зрения, оба процесса описываются следующим выражением:

$$H(X_n)^T = 0 \quad где\ X_n = X_1, X_2, \ldots, X_N$$

$$(X_n)^T = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix}$$

X_i — символ кодового слова, старший символ имеет наименьший индекс; H — это матрица размера [r*n], причем эта матрица H представлена в виде двух подматриц A и I.

A=[r*k]

I=[r*r]

Кодирование сообщения всегда должно удовлетворять соотношению $H(X_n)^T = 0$. Причем вес столбцов матрицы A всегда больше либо равен 2:

$$W(H_i)_A \geq 2$$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Как видно для матрицы I вес вектор столбцов, как и вектор строк всегда равен 1.

Подматрицу A можно определить как:

$$A = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ h_{r1} & \cdots & \cdots & h_{rk} \end{bmatrix}$$

Элемент h_{ij} это один из двух бинарных символов и $i = \overline{1,r} \quad j = \overline{1,k}$

В этом случае код простой честности можно представить как матрица, содержащая одну строку из единиц:

$$H_{nk} = \left[\underbrace{11111}_{k} \ldots \underbrace{1}_{\Sigma} \right]$$

Это означает, что избыточный элемент будет равен:

$$X_{r1} = \sum_{j=1}^{k} X_j$$

А в соответствии с математическим выражением кодирования информации количество столбцов подматрицы I всегда равно числу избыточных символов, количество столбцов подматрицы А — количество информационных символов. Общее количество столбцов матрицы Н равно длине кодового слова.

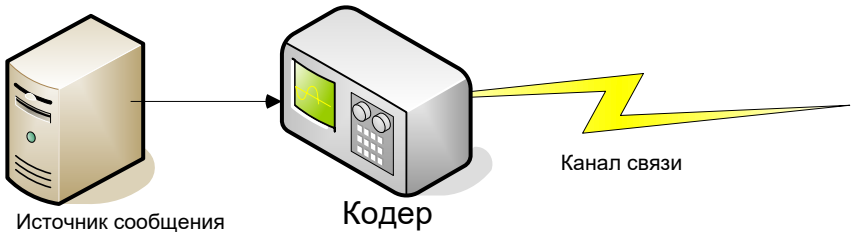
Рассмотрим пример. Пусть ситуация обратная, X_k=1, т.е. длина информационного слова k=1 бит. Предположим что r=5, n=k+r=6. Это означает, что относительная скорость передачи информации при использовании такого кода равна R=k/n=1/6, т.е. скорость передачи информации при использовании такого кода ниже в 6 раз, чем без него. Относительная избыточность $R_и$=r/k=5, то есть избыточность в 4 раз больше информационного слова.

$$H_{6,1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ \underbrace{1}_{A} & \underbrace{0 & 0 & 0 & 0 & 1}_{I} \end{bmatrix}$$

Подматрица А имеет размер 5 строк на 1 столбец, подматрица I — 5строк на 5 столбцов. В соответствии с $H(X_n)^T = 0$ и составленной матрицей, первый проверочный символ будет повторением информационного слова $X_{r,1}=X_1$, второй проверочный символ — $X_{r,2}=X_1$, третий — $X_{r,3}=X_1$ и т.д. Имеющийся информационный символ будет повторен 6 раз и в соответствии с этим, кодовое слово $X_n = \underbrace{1}_{X_k}\underbrace{11111}_{X_r}$

Если же взять X_k=0, то при r=5 матрица меняться не будет, то есть не зависит от информационного слова. $X_n = \underbrace{0}_{X_k}\underbrace{00000}_{X_r}$

Таким образом, ясно, что любой помехоустойчивый код использует выражение $H(X_n)^T = 0$ для формирования и проверки передаваемой информации на наличие ошибок. На передающей стороне имеем источник сообщения и кодер, работающий на основании математического выражения для помехоустойчивого кодирования.



Декодер работает на принципе сравнения принятого дополнительного символа (-ов) и вычисленного на принимающей стороне.

$Y_n \to \hat{X}_k$ Если в принятом сообщении нет ошибок, то слово Y_n точно соответствует слову X_k, однако, если произошла ошибка, то Y_n обязательно отличается от X_k.

$Y_k \neq X_k$ – *если ошибка произошла среди информационных символов*

$Y_r \neq X_r$ – *если ошибка в проверочных символах*

$Y_k \neq X_k, Y_r \neq X_r$ – *если ошибка в обеих частях*

Теперь, допустим, использовалось информационное слово X_k=1111, k=4, r=1.

$$H_{5,4} = \left[\underbrace{1111}_{A}\underbrace{1}_{I}\right] \Rightarrow [11111] \bullet \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow 0$$

Получаем $X_n = \underbrace{1111}_{X_k}\underbrace{0}_{X_r}$

При выборе кода для помехоустойчивого кодирования надо в первую очередь ориентироваться на характеристики канала связи и возможные помехи, т.е. код должен с большей вероятностью выявлять более вероятные типы ошибок.

К примеру, если сравнить две матрицы:

$H_{5,4} = [1111 \quad 1] \Rightarrow X_r = 1 \oplus 1 \oplus 1 \oplus 1$

$H_{5,4} = [1001 \quad 1] \Rightarrow X_r = 1 \oplus 1$

То сразу видно, что во втором варианте выполняется меньше математических действий, что означает сокращение времени кодирования.

Пусть X_k=1001, k=4. Считаем, что ошибки равновероятны во всех символах, однако r=1 (используем 1 проверочный символ).

$H_{5,4} = [11111]$

$X_r = \sum_{j=1}^{n} X_j \, (\textit{по модулю } 2) = 1 \oplus 0 \oplus 0 \oplus 1 = 0$

$X_n = X_5 = \underbrace{1001}_{X_k} \underbrace{0}_{X_r}$

При t=0 $Y_n = \underbrace{1001}_{Y_k} \underbrace{0}_{Y_r}$, т.е. $X_n=Y_n$ что означает что ошибок нет.

При t>0:

а) $Y_n = \underbrace{0001}_{Y_k} \underbrace{0}_{Y_r}$ в этой ситуации $Y_k \neq X_k$, *а* $Y_r = X_r$

б) $Y_n = \underbrace{1001}_{Y_k} \underbrace{1}_{Y_r}$ в этой ситуации $Y_k = X_k$, *а* $Y_r =\neq X_r$ ошибка в доп. символах.

Декодирование кодовых слов. Поиск и исправление ошибок.

Задача приемной стороны — анализ принятой последовательности. Этот процесс опирается на следующий математический факт:

$H(Y_n)^T = 0$

Это соотношение с точностью до множителя повторяет соотношение для кодирования, эти два соотношения основаны на использовании одной и той же матрицы H. Если $X_n \equiv Y_n$ то два этих соотношения идентичны, если же $X_n \neq Y_n$ то Y_n можно представить как $Y_n = X_n + E$, где E — двоичный вектор, вектор ошибки.

К примеру:
$X_n = 1001\,0 \quad Y_n = 00010 \Rightarrow E = 10000$
$X_n = 1001\,0 \quad Y_n = 11101 \Rightarrow E = 01111$

Вес — количество ошибок, появившихся в процессе передачи с учетом вектора E.

$H(X_n + E)^T = 0$

$\underbrace{H(X_n)^T}_{0} + \underbrace{H(E)^T}_{S} = 0$ где S — значение синдрома

Показателем качества передачи информации является r-разрядный двоичный вектор S, который называется **синдромом ошибки.**

В соответствии с формулами, приведенными выше, если ошибок нет (t=0), то и синдром должен быть тождественно равен нулю. Т.о. если вес синдрома равен нулю, то ошибок нет.

$t \to 0 \Rightarrow S \to 00\ldots0 \Rightarrow W(S) = 0$

Если же произошла одиночная ошибка то синдром в соответствии с формулами равен вектор столбцу подматрицы A или I.

Практический алгоритм вычисления синдрома

1. Используя полученный символы Y_k, на основе заданной матрицы H_{nk}, вычисляют проверочный символ Y_n. $Y_k \xrightarrow{H_{nk}} Y_r'$
2. $S = Y_k \oplus Y_r'$ — вычисляем синдром
3. Анализируем синдром и в соответствии с результатом анализа предпринимаем последующие действия.

В зависимости от корректирующей способности кода, после анализа синдрома при наличии ошибок принимается решение об их исправлении, если позволяет код, либо о повторной передаче информации, если код не позволяет исправить ошибку. Формально корректирующие способности кода определяются минимальным кодовым расстоянием (см. выше).

Код Хэмминга

Код Хэмминга с минимальным кодовым расстоянием $d_{min}=3$

Код Хэмминга с $d_{min}=3$ это наиболее простой вид линейного кода, который позволяет не только обнаруживать, но и исправлять ошибки.

$t_0=1$ — Single Error Detection

$t_n=1$ — Single Error Correction

Линейным называют такой код, в котором если X_n и — кодовый слова принадлежащие одному кодовому пространству, то X_n+Y_n является кодовым слово принадлежащим тому же кодовому пространству. Алгоритм обнаружения и исправления ошибок основан на анализе синдрома. Кодирование информации основано на использовании соотношения $H(X_n)^T = 0$ и $H(Y_n)^T = 0$ где $Y_n = X_n + E$.

В коде Хэмминга r>1 и $n \to \{X_n\}$

Должно быть n различных синдромов, соответствующих одиночным ошибкам:

$n \to S_r \neq 0$ (t=1).

Также должен быть один синдром, при котором ошибок нет:

$n \to S_r \equiv 0$ (t=0).

r — $\{S_r\}$ — r-разрядный двоичный вектор S_r

Общее число комбинаций 2^r должно быть, по крайней мере, $2^r \geq n+1$ в силу того, что:

$$2^r \geq n+1 \Rightarrow 2^r \geq k+r+1 \Rightarrow r \geq \log_2(k+r+1) \Rightarrow$$

$$r \geq \log_2(k+1)$$

В коде Хэмминга с минимальным кодовым расстоянием $d_{min}=3$ проверочная матрица H имеет классический вид и состоит из двух подматриц размером k*r и r*r.

$$\underbrace{H}_{n*r} = \left[\underbrace{A}_{r*k} \ \underbrace{I}_{r*r} \right]$$

Минимальный вес каждого из столбцов матрицы H должен быть ≥ 2.

Рассмотрим пример:

Имеется информационная основа X_k=1001

k=4

r≥$\log_2(4+1)$=3

n=k+r=7

Любой код задается тремя параметрами — n,k и d.

$$\underbrace{H_{7,4}}_{n*k} = \left[\begin{array}{ccccccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \underbrace{1 & 1 & 0 & 1}_{A} & \underbrace{0 & 0 & 1}_{I} \end{array} \right]$$

Вычисление проверочных символов:

$$X_{r,j} = \sum_{\substack{j=1 \\ i=1}}^{k,r} (\textit{по модулю } 2) h_{j,i} * X_i \ \text{ где } \ i = \overline{1,r} \ \ j = \overline{1,k}$$

В соответствии с этим первый проверочный символ:

$$X_{r,1} = h_{11} * X_1 \oplus h_{12} * X_2 \oplus h_{13} * X_3 \oplus h_{14} * X_4 = 0*1 \oplus 1*0 \oplus 1*0 \oplus 1*1 = 1$$

$$X_{r2} = 0 \quad X_{r3} = 0 \Rightarrow X_r = 100$$

$$X_n = X_k * X_r = 1001100$$

1. Рассмотрим ситуацию, когда ошибок в переданной информации нет, t=0:

Y_n=1001 100 Y_n=X_n t=0

Вычислим новый набор проверочных символов:

$$\{Y_r'\} = \sum_{\substack{i=1 \\ j=1}}^{r,k} (\textit{по модулю } 2) h_{j,i} * Y_i$$

$$Y_r' = 100 \quad X_r \equiv Y_r'$$

Синдром равен:

$$S = Y_r + Y_r' = 100 \oplus 100 = 000 \equiv 0$$

Нулевой синдром означает безошибочную передачу или прием информации. Вес синдрома равен нулю, что означает правильность передачи.

2. Рассмотрим ситуацию, когда возникает одиночная ошибка, t=1:

- Пусть ошибка произошла в служебных символах $Y_n = 1001\underline{0}00$

Вес синдрома равный 1 означает, что ошибка произошла среди служебных (избыточных) символов. Из соотношения видно:

$$H(Y_n)^T = 0$$

$$Y_n = X_n + E \quad E_7 = 0000100$$

$$H(E_n)^T = 100$$

Синдром соответствует вектор столбцу матрицы H, в котором произошла ошибка. Синдром соответствует вектор столбцу, вес которого равен 1 биту.

В общем случае, если использовать стандартный алгоритм, исправление ошибки основывается на том, что к полученному слову Y_n добавляют вектор ошибки E_n.

$$\hat{X}_n = Y_n \oplus E_n$$

$$1001000 \oplus 0000100 = 1001100 \equiv X_n$$

- Пусть ошибка произошла в бите информационного слова $Y_n = \underline{0}001100$

Вычислим дополнительные проверочные символы

$$Y_r' = 111$$

$$S = Y_r \oplus Y_r' = 011$$

S≠0 — есть ошибка и она обнаружена

Вес синдрома >1 значит, что ошибка произошла в информационных битах.

011 — первый вектор столбец матрицы A => E=1000000

0001100⊕1000000=1001100≡X_n

Код Хэмминга с d_{min}=3 гарантированно **обнаруживает и исправляет одиночную ошибку** в любом разряде кодового слова.

Местоположение вектор столбцов в матрице A не имеет особого значения однако на передающей и принимающей сторонах эти матрицы **должны быть идентичны.**

В случае возникновения двух ошибок (t=2), в силу соотношения $H(X_n + E)^T = 0$ W(E_n)=2 и синдром будет равен сумме вектор столбцов по модулю 2 что не позволяет правильно идентифицировать место возникновения ошибки.

Пример кодирования сообщения X_k=101011101

$$k = 9 \quad r \geq \log_2 9 + 1 = 5 \quad n = 14$$

$$H_{14,9} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

$$\underbrace{1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1}_{X_k}\ \underbrace{0\ 0\ 1\ 1\ 0}_{X_r}$$

$$X_r = h_{11} * X_1 \oplus h_{12} * X_2 \oplus \dots$$

Использование Turbo кодов в системах передачи информации

Перейти

Код Хэмминга с минимальным кодовым расстоянием $d_{min}=4$

Этот код позволяет **обнаруживать 2 ошибки** и гарантированно **исправлять одну**. Алгоритм кода такой же как и для кода Хэмминга с $d_{min}=3$, основное отличие — дополненная поверочная матрица H. Для преобразования ее дополняют строкой, позволяющей вычислить сумму по модулю 2 всех символов кодового слова X_n для случая с $d_{min}=3$. Т.е. получаем добавочную строку состоящую из единиц. Таким образом, количество избыточных символов в коде с $d_{min}=4$ есть $r_{d=3}+1=r_{d=4}$ и мы получаем трансформированную матрицу:

$$H_{d=3} = [A \vdots I] \Rightarrow \begin{bmatrix} A' & I' \\ \underbrace{r' \times k \quad r' \times r'}_{r' \times n'} \end{bmatrix} \quad \begin{matrix} r' = r_{d=4} \\ n' = n+1 \end{matrix}$$

Запись матрицы в таком виде не является канонической, т.к. подматрица не единичная диагональная матрица. Для преобразования такой записи к каноническому виду воспользуемся свойствами линейного кода.

Свойства матрицы не изменятся, если в последней строке записать сумму по модулю 2 всех строк подматриц A и I. Полученная матрица будет является проверочной для кода с dmin=4.

После преобразования **вес** каждого вектор столбца будет **нечетным**, Это основное отличие канонической матрицы от исходной (d=3).

Рассмотрим пример:

Пусть k=6 для случая d=3

$r=\log_2 k+1=4$, т.о. n=10

При этих параметрах матрица $H_{10,6}$ может иметь следующий вид:

$$H_{10,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Стоит задача преобразования такой матрицы для случая d=4. Дополним ее строкой состоящей из единиц:

$$H_{10,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

Для преобразования такой матрицы к каноническому виду надо сложить по модулю 2 каждый из столбцов матрицы H, и записать полученное выражение в последнюю стоку этого столбца. Также надо дополнить матрицу I дополнительным столбцом.

$$H_{10,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Любая проверочная матрица коды Хэмминга с минимальным кодовым расстоянием $d_{min}=4$ имеет нечетный вес столбцов, т.е. вес любого из столбцом подматрицы A` может быть равен 3,5,7,… а вес столбцов матрицы I` всегда равен 1, поскольку это единичная подматрица.

Как и в предыдущем случае, **равенство нулю синдрома** означает **отсутствие ошибок**. Если же синдром **не равен нулю** и имеет **нечетный вес**, то это означает что произошла **одиночная ошибка**, т.к. синдром одиночной ошибки всегда равен столбцу подматрицы, в котором произошла ошибка. Если же синдром **не равен нулю** и его **вес четный**, то произошла **двойная ошибка**, т.к. вес суммы любых двух столбцов всегда четный.

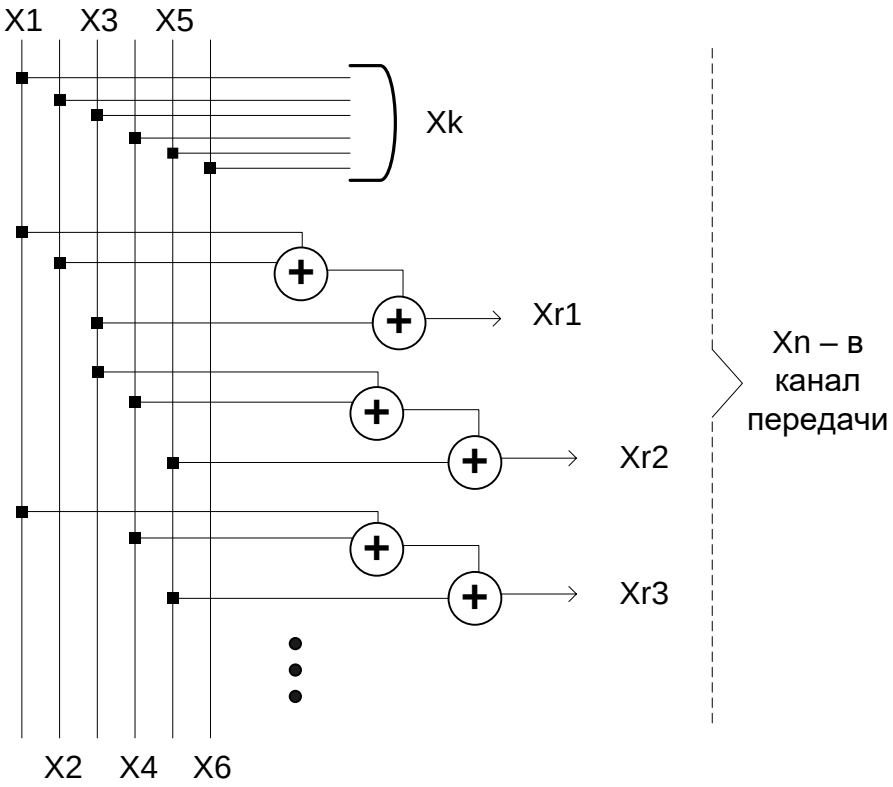
Коррекция одиночной ошибки аналогична коду Хэмминга с $d_{min}=3$. Скорректировать же двойную ошибку не представляется возможным, т.к. синдром с четным весом может соответствовать любой комбинации двух ошибок.

Аппаратная реализация кодера и декодера.

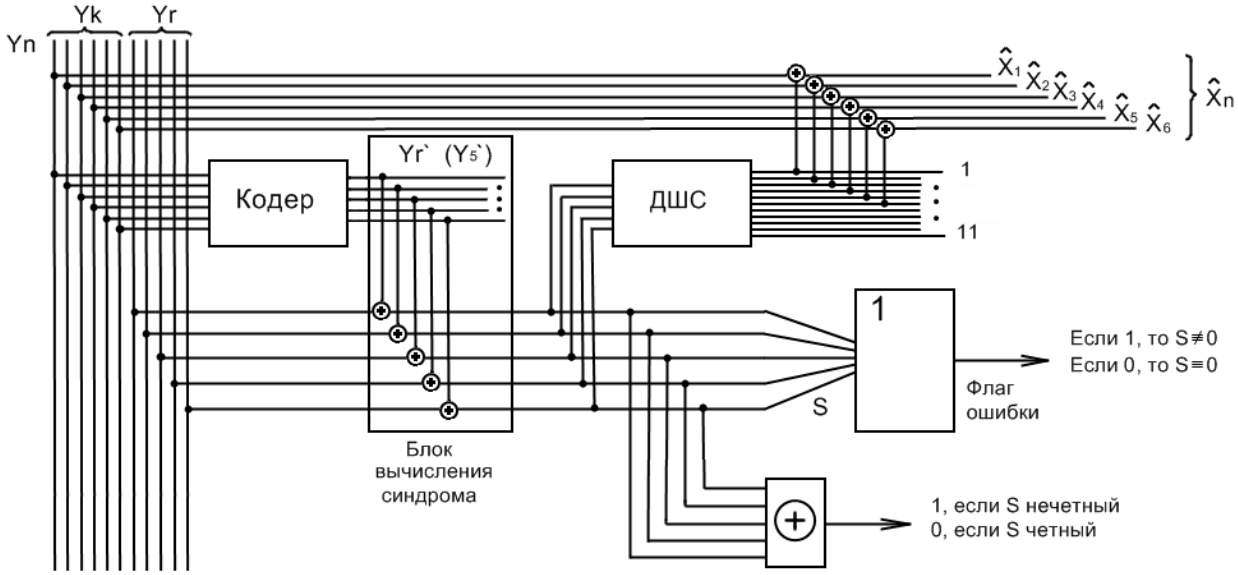
Рассмотрим устройство кодера и декодера для случаев канонического и не канонического вида матрицы H.

Внутренняя структура оборудования на передающей и принимающей стороне определяется в основном используемыми проверочными матрицами A и I. Эти устройства могут быть построены на основе дискретных логических элементов, либо на основе программируемых логических матриц. Программируемые матрицы — это соответствующим образом собранные логические элементы, которые программируются на определенную матрицу посредством расплава плавких предохранителей.

На передающей стороне **кодер** для k=6 выглядит примерно так:



На принимающей стороне, схема **декодер**а кода Хэмминга выглядит следующим образом:



Если синдром не равен нулю, то на выходе схемы получаем 1, при этом далее необходимо определить одиночная или двойная ошибка произошла. После определения количества ошибок, если произошла двойная — посылаем запрос на

повторную пересылку, если же одиночная — в дешифраторе простым сравнением определяем в какой позиции ошибка и исправляем ее.

Другие помехоустойчивые коды

Составной код. Итерационные коды.

Итерационный код обладает теми же свойствами что и код Хэмминга, является линейным, в наиболее простом случае позволяет корректировать одиночные ошибки. Использование данного кода определяется записью проверочных символов в идее **двумерной матрицы**.

К примеру, имеем информационное слово X_k=10110111. Это слово можно записать в виде двумерной матрицы:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$ Проверочные символы вычисляются отдельно по строкам и столбцам.

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} 1 \\ 1 \end{matrix} \} X$$

$$\underbrace{1 \quad 1 \quad 0 \quad 0}_{Y}$$

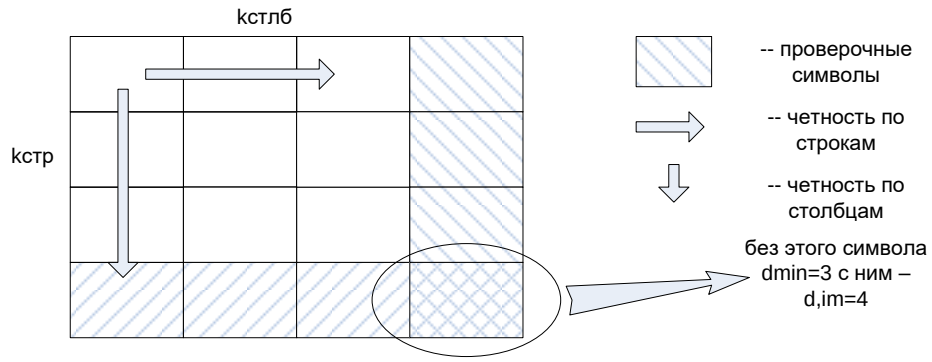
Если k=Xстр*Yстлб, то r=X+Y

Вычисление этих проверочных символов также можно представить в виде классической проверочной матрицы.

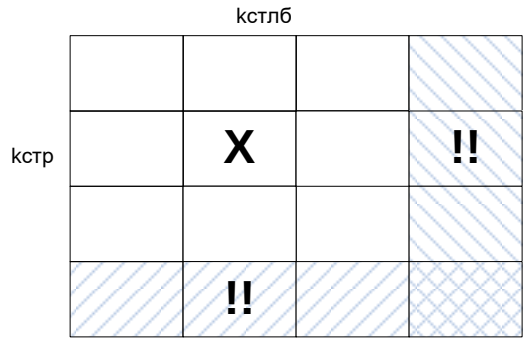
$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Модифицированный аддитивный код

В общем случае информационное слово мы записываем в виде матрицы:



Принцип обнаружения одиночной ошибки основан на изменении четности по строкам и столбцам. Если X — ошибка то изменение 2-х проверочных символов покажут место где произошла ошибка.



Пример: X_k=101010101

При появлении ошибки проверочные символы сразу указывают место ее возникновения.

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 0 \end{matrix}$$

$$0 \quad 1 \quad 0 \ (0)$$

$$X_n = \underbrace{101010101}_{X_k} \underbrace{010010}_{X_r}$$

$$X_n = \underbrace{111010101}_{X_k}\ \underbrace{010010}_{X_r}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ 1 \\ 0 \end{matrix}$$

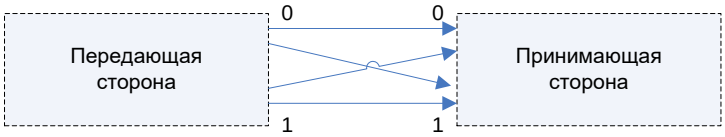
1 0 0 (0)

При такой записи **исправляются все одиночные ошибки**, также при использовании дополнительного символа **детектируются двойные ошибки**.

На основании итерационных кодов строятся турбо коды, где контроль четности осуществляется по диагонали. Такие дополнительные символы позволяют устранить неоднозначности при появлении двойных и более ошибок. См. реферат.

Характеристика надежности двоичного канала передачи при использовании кодов

Возьмем для простоты симметричный канал передачи данных.



Двоичный симметричный канал связи характеризуется тем, что вероятность:

$$p(y_i = 1 \mid x_i = 0) = p(y_i = 0 \mid x_i = 1)$$ **Основная вероятностная характеристика** двоичного канала передачи. С равной вероятностью ошибка может произойти в произвольном символе кода.

p(110)=p(011)=p — условная вероятность

p(1|0)+p(0|1)=1

На практике, однако, каналы характеризуются различными вероятностями, т.е.:

$$p(y_i = 1 \mid x_i = 0) \neq p(y_i = 0 \mid x_i = 1)$$

Т.к. вероятностей преобразования 0 -> 1 и 1 -> 0 существует бесконечное множество, то существует и бесконечное множество различных Z-каналов — каналов с преобразованием ошибок.

Качество канала оценивается по вероятности полного совпадения переданного и полученного сообщения (p(x=y)=1). Возможна оценка канала по двум типам передачи:

1. Без кодирования, т.е. передается слово X_k [$p(X_k=Y_k)=p(t=0)=p^0(1-p)^k=(1-p)^k$]
2. С кодированием, свойства канала не изменяются. Однако, если рассмотреть случай передачи кода корректирующего одиночную ошибку:

$$p(t=0)=(1-p)^n$$

$$p(t=1)=\binom{n}{1}*p^1*(1-p)^{n-1} \qquad где \binom{n}{1} \text{ числосочетаний } n \text{ по } 1$$

В общем случае, так как ошибка с равной вероятностью может произойти в любом из символов, то общее число комбинаций вычисляется как:

$$i \to \binom{n}{i}=\frac{n!}{i!*(n-i)!} \Rightarrow p(t=1)=n*p(1-p)^{n-1}$$

$$p(X_n=\hat{Y}_n)=(1-p)^n+n*(1-p)^{n-1}$$

Сравнивая вероятности без кодирования и с использованием кодирования можно оценить **эффективность использования кода**.

Пример:

В реальных каналах передачи $p=10^{-3}\ldots10^{-7}$. Пусть в нашем канале связи $p=10^{-3}$, k=512 bit. Используется простой код корректирующий одиночную ошибку. Оценим эффективность кода.

В соответствии с формулой для канала без использования кода $p(X_k=Y_k)=(1\text{-}0{,}001)^{512}$.

При использовании кода $r=\log_2 512+1=10$ => n=522. Получаем $p(X_n=\hat{Y}_n)=(1-0.001)^{522}+522*0.001*(1-0.001)^{521}$

Т.о., сравнивая эти полученные результаты, мы получаем качественную оценку эффективности использования кода.

Суммируя все вышесказанное по первому методу преобразования информации:

- Всегда приводит к удлинению передаваемой последовательности n>k
- Главное назначение — повышение уровня надежности канала передачи или устройства хранения информации.

Преобразование информации на основе методов сжатия (компрессии)

Методы компрессии информации преследуют 3 основных цели:

1. Снижение физического объема хранимой на носителях информации.
2. Снижение стоимости передачи фиксированного объема информации.
3. Повышение уровня конфиденциальности информации.

Основной особенностью методов является то, что количество передаваемой информации меньше чем до преобразования n<k.

Общая классификация методов сжатия:

- **Символ-ориентированные методы**. Основаны на поиске и анализе повторяющихся символов (комбинаций) и их замене на другие комбинации меньшей длины.
- **Вероятностные методы**. Основаны на анализе частотных характеристик используемого алфавита.
- **Комбинированные методы**. Объединяют первые и вторые, что может давать новые качественные свойства.

Также методы сжатия можно разделить на группы по потерям информации после декомпрессии:

- **Сжатие без потерь данных**. Методы сжатия при которых объем и вид информации до сжатия и после декомпрессии идентичны, нет изменений вызванных реализацией алгоритма. (текст, файлы БД, программы, другие виды информации где недопустимы потери)
- **Сжатие с возможной потерей информации**. Объединяет методы, при которых исходная и декомпрессированная информация может не совпадать. (файлы мультимедиа, звук, видео, изображения)

Характеристики методов сжатия

Основной численной характеристикой методов сжатия является коэффициент сжатия R. Коэффициент сжатия считают по двум основным методам:

- $R_I = \frac{V_{после\_сжатия}}{V_{до\_сжатия}}$ $V_{после\_сжатия}$ *– объем информации после сжатия*, $V_{до\_сжатия}$ *– объем информации до сжатия*
- $R_{II} = \frac{V_{до\_сжатия} - V_{после\_сжатия}}{V_{до\_сжатия}} = 1 - \frac{V_{после\_сжатия}}{V_{до\_сжатия}}$

На практике чаще используют вторую формулу.

Второй численной характеристикой является интегральный коэффициент сжатия (смотри ниже).

Символьные методы сжатия

Наиболее простым из символьных методов является метод **интервалов**. Суть метода в поиске одинаковых комбинаций символов и замене их на специальную комбинацию abc, где a — специальный символ, b — один из символов выбранной повторяющейся последовательности (интервал) и c — количество повторений.

Пример:

X_k = mkkkcb0000fkff

Обычно интервалом считают последовательность не менее двух повторяющихся символов, значит, в нашем примере можно выделить 3 интервала:

Xk = mkkkcb0000fkff

При реализации такого метода сжатия важным вопросом является выбор специального символа a. Его выбирают так, чтобы вероятность его появления в тексте стремилась к минимуму. Для примера возьмем специальный символ равным ‘*’.

После преобразования (сжатия) получим такую последовательность:

H_k = m*k3cb*04fk*f2

Как видно из примера метод является эффективным при количестве повторений не ниже трех. Если посчитать коэффициент сжатия $R_{II} = 1 - \frac{15}{16} = \frac{1}{16}$

Метод Burrows-Wheller

Суть метода в том, что исходная последовательность разбивается на блоки одинаковой длины и преобразованию подвергается каждый блок в отдельности. На каждом этапе создается квадратная матрица W_1 размера k*k, где k — длина блока, причем каждая последующая строка, начиная со второй, является циклическим сдвигом на один символ влево предыдущей строки. На основе первой матрицы создается вторая матрица W_2 размера k*k содержащая строки первой матрицы, отсортированные в лексикографическом порядке. Результатом преобразования является столбец h_k матрицы W_2 и число N соответствующее номеру строки матрицы W_2 в которой записана исходная последовательность.

Обратное преобразование основано на свойстве рекуррентности преобразования матрицы W_1 и W_2.

Пример:

Пусть X_k=столб

$$W_1 = \begin{vmatrix} с & т & о & л & б \\ т & о & л & б & с \\ о & л & б & с & т \\ л & б & с & т & о \\ б & с & т & о & л \end{vmatrix} \Rightarrow W_2 = \begin{vmatrix} б & с & т & о & л \\ л & б & с & т & о \\ о & л & б & с & т \\ с & т & о & л & б \\ т & о & л & б & с \end{vmatrix}$$

h_5=лотбс, N=4

Информация после преобразования (условной компрессии) имеет больший объем чем до, в силу этого метод не используется в одиночку, однако в сочетании с другими методами (особенно при сжатии двоичных последовательностей) очень эффективен.

Обратная процедура:

Y_k=лотбс N=4

$$W = \begin{vmatrix} л \\ о \\ т \\ б \\ с \end{vmatrix} \Rightarrow \begin{vmatrix} л & б \\ о & л \\ т & о \\ б & с \\ с & т \end{vmatrix} \Rightarrow \begin{vmatrix} л & б & с \\ о & л & б \\ т & о & л \\ б & с & т \\ с & т & о \end{vmatrix} \Rightarrow$$

$$\Rightarrow \begin{vmatrix} л & б & с & т \\ о & л & б & с \\ т & о & л & б \\ б & с & т & о \\ с & т & о & л \end{vmatrix} \Rightarrow \begin{vmatrix} л & б & с & т & о \\ о & л & б & с & т \\ т & о & л & б & с \\ б & с & т & о & л \\ с & т & о & л & б \end{vmatrix} \Rightarrow W = \begin{vmatrix} б & с & т & о & л \\ л & б & с & т & о \\ о & л & б & с & т \\ с & т & о & л & б \\ т & о & л & б & с \end{vmatrix}$$

На каждой итерации происходит дополнение матрицы исходной последовательностью и лексикографическая сортировка строк матрицы. В результате получаем исходное слово X_k=столб.

Статистические методы сжатия

Статистические методы используют вероятностные характеристики для построения стратегии сжатия. Среди этих методов наиболее известны 2:

1. Метода Шеннона-Фано
2. Метод Хаффмана

Метод Шеннона-Фано

Суть метода: отсортированные в последовательности от Max к min вероятности появления всех символов алфавита в тексте, делят на 2 части в пропорции, чтобы сумма вероятностей в каждой из частей была по возможности близка (0,5:0,5). Символам верхней части приписывают старший символ кода (единица 1), нижней части — младший символ (нуль 0). В процессе компрессии эти коды будут заменять исходные коды символов (ASCII коды). Далее, каждую

из частей, в свою очередь, опять делят на две до тех пор, пока в результате разделения каждая из частей не будет содержать одного символа.

К примеру:

Имеем алфавит A:a_i[i=1…10] => N(A)=10 (мощность)

$p(a_1) = 0.05 \quad p(a_6) = 0.12$

$p(a_2) = 0.10 \quad p(a_7) = 0.05$

$p(a_3) = 0.03 \quad p(a_8) = 0.10$

$p(a_4) = 0.20 \quad p(a_9) = 0.15$

$p(a_5) = 0.15 \quad p(a_{10}) = 0.05$

Отсортируем вероятности появления по убыванию и разделим на группы по вероятности, формируя при этом коды заменяющие символ:

$p(a_4) = 0.20 \quad 11$

$p(a_5) = 0.15 \quad 101$

$p(a_9) = 0.15 \quad 100$

$p(a_6) = 0.12 \quad 011$

$p(a_2) = 0.10 \quad 010$

$p(a_8) = 0.10 \quad 0011$

$p(a_1) = 0.05 \quad 0010$

$p(a_7) = 0.05 \quad 0001$

$p(a_{10}) = 0.05 \quad 00000$

$p(a_3) = 0.03 \quad 00001$

Получили таблицу кодировки:

$p(a_1) = 0010 \qquad p(a_6) = 011$

$p(a_2) = 010 \qquad p(a_7) = 0001$

$p(a_3) = 00001 \quad p(a_8) = 0011$

$p(a_4) = 11 \qquad p(a_9) = 100$

$p(a_5) = 101 \qquad p(a_{10}) = 00000$

Эту таблицу должен иметь и компрессор и декомпрессор. Пусть сообщение имеет вид: $a_5 a_4 a_4 a_9 a_6 a_1 a_1 a_8 a_7 a_9 \Rightarrow 10111111000110010001000110001100$ получили сжатое сообщение n=32 bit.

При использовании таблицы кодировки определенной длины, к примеру, ASCII, наша последовательность занимала бы 8*10=80 бит, после сжатия мы получили 32 бит.

Обратное преобразование:

Для преобразования необходимо знать таблицу кодировки, l_{min}, l_{Max}, где l_{min} и l_{Max} — минимальная и максимальная длина кода в битах соответственно. Необходим буфер длиной равный l_{Max}. Для примера приведенного выше, $l_{min} = 2$, $l_{Max} = 5$

Декомпрессия происходит в несколько стадий:

1. Заносим в буфер l_{min} символов и сравниваем содержимое буфера с таблицей кодировки. $\underrightarrow{Y_n}$ _ _ _ 1 0
2. Если не найдено соответствие, то сдвиг на 1 символ влево и опять сравнение с таблицей. $\underrightarrow{Y_n}$ _ _ 1 0 $1 \rightarrow a_5$ Если совпадение найдено, то обнуляем буфер и записываем соответствующий символ.
3. Заносим в буфер l_{min} символов и сравниваем содержимое буфера с таблицей кодировки. $\underrightarrow{Y_n}$ _ _ _ 1 $1 \rightarrow a_4$
4. …
5. Если символов во входной последовательности не осталось — декомпрессия закончена.

Характеристикой эффективности алгоритма является так называемый интегральный коэффициент компрессии:

$C = \sum_{i=1}^{n} p(i) * l_i$ где l_i — количество символов кода соответствующее символу алфавита a_i

В нашем примере $C = \underset{a_1}{0.05} * 4 + \underset{a_2}{0.10} * 3 + \underset{a_3}{0.03} * 5 + \ldots + \underset{a_{10}}{0.05} * 5$

На практике этот коэффициент означает среднее количество символов кода, соответствующее одному символу алфавита.

Выводы:

- Если алфавит достаточно большой, то можно утверждать о существовании **нескольких вариантов разделения** массива на подмассивы. Наилучшим является тот, которому соответствует min значение C.

- Важнейшим звеном алгоритма является составление кодера. Правильное составление основано на **требовании префикса**, которое означает, что произвольная бинарная комбинация меньшей длины не должна быть началом большей комбинации.

Также, для такого кода важным является повышение **надежности передачи**. Т.к. при появлении ошибки в принятой последовательности возможна ситуация когда не будет соответствия между кодом и таблицей кодировки.

Метод Хаффмана.

Классический метод, основанный на априорной вероятности появления всех символов в тексте.

Сущность алгоритма:

1. как и в предыдущем методе следует отсортировать символы от максимума до минимума.
2. далее строится дерево объединением попарно символов алфавита с примерно одинаковыми вероятностями, начиная от пары с наименьшей вероятностью. Объединенные символы создают новый “виртуальный символ” с априорной вероятностью равной сумме вероятностей объединенных символов.
3. объединение символов таким образом, создает ветви и узлы дерева. Каждая из ветвей должна быть обозначена бинарным символом. Обычно это делается так: меньшей вероятности приписывается 0, большей вероятности из пары — 1. Последний из объединенных символов создает корень дерева.
4. после построения дерева каждому из символов исходного алфавита приписывается соответствующий код, который получается обходом ветвей дерева от корня к этому символу.
5. сжатие и декомпрессия исходного сообщения и соответствующего кода могут производиться, как и ранее (для Шеннона).

Рассмотрим на примере метод реализации Хаффмана.

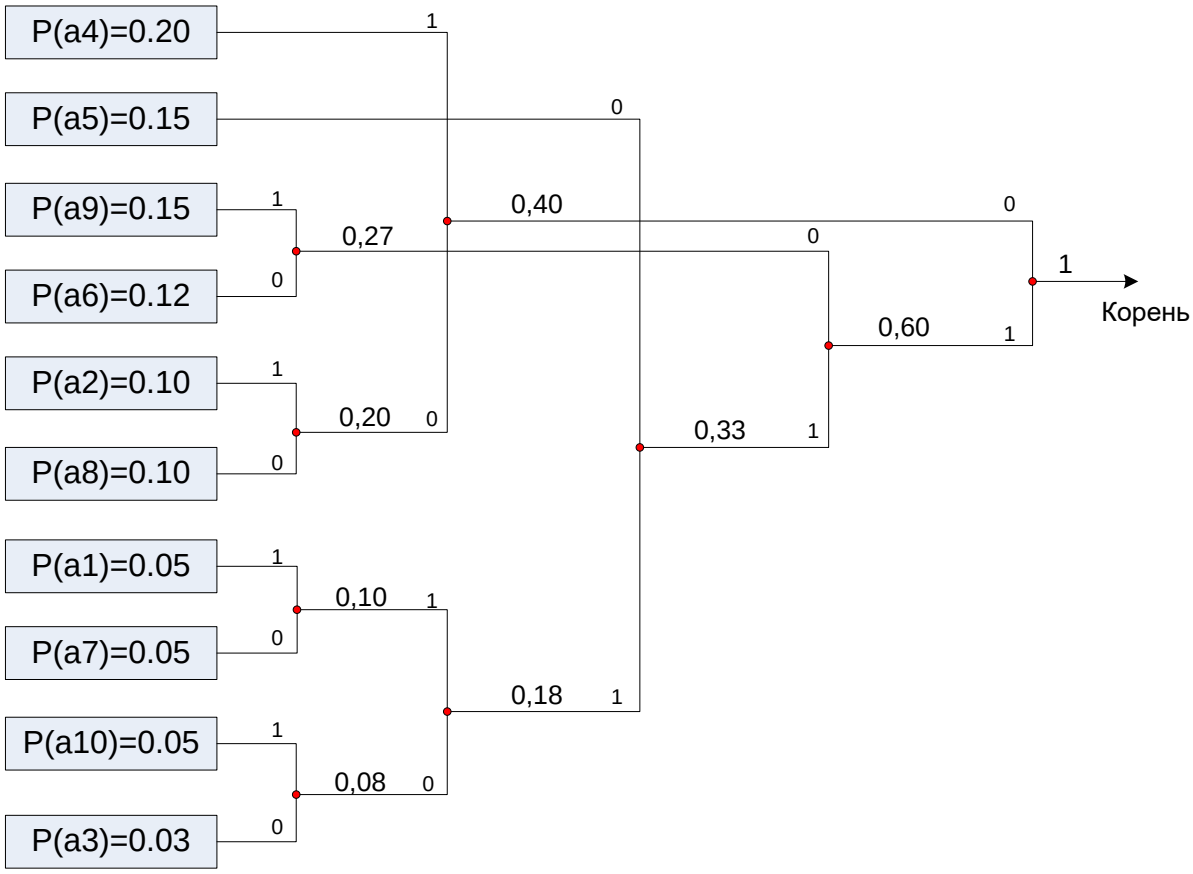
$A \{a_i\}$;
$i \in [1,N]$, где $N = 10$.
$p(a_1) = 0.05$;
$p(a_2) = 0.10$;
$p(a_3) = 0.03$;
$p(a_4) = 0.20$;
$p(a_5) = 0.15$;
$p(a_6) = 0.12$;
$p(a_7) = 0.05$;
$p(a_8) = 0.10$;
$p(a_9) = 0.15$;
$p(a_{10}) = 0.05$;

Сортируем от максимума до минимума:

$p(a_4) = 0.20$

$p(a_5) = 0.15$
$p(a_9) = 0.15$
$p(a_6) = 0.12$
$p(a_2) = 0.10$
$p(a_8) = 0.10$
$p(a_1) = 0.05$
$p(a_7) = 0.05$
$p(a_{10}) = 0.05$
$p(a_3) = 0.03$

Объединяем попарно символы:



Запишем соответствующие символам коды, посредством обхода ветвей дерева от корня к этому символу:

$a_1 = 11111$
$a_2 = 001$
$a_3 = 11100$

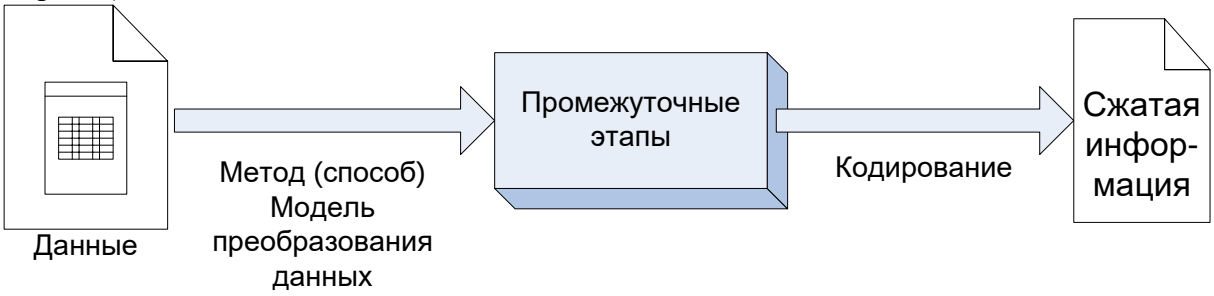
a_4 = 01
a_5 = 110
a_6 = 100
a_7 = 11110
a_8 = 000
a_9 = 101
a_{10} = 11101

Выводы:

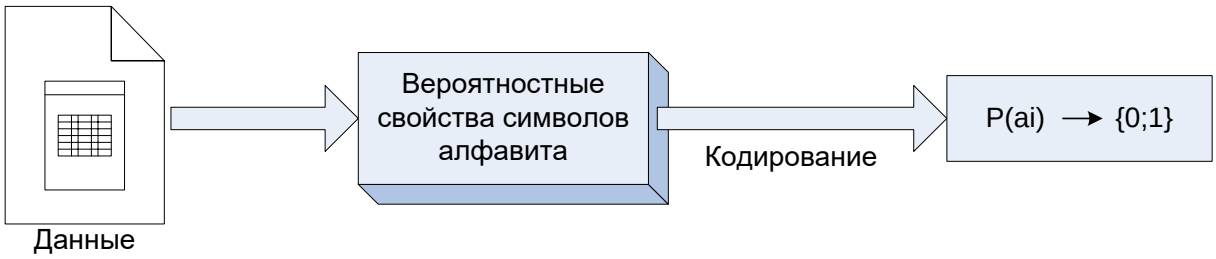
Методы сжатия основаны на замене общих строк знаков или последовательно встречающихся одинаковых знаков другими зарезервированными для этого знаками. Эти методы могут давать некоторый небольшой уровень безопасности в дополнение к экономии памяти. Чтобы сэкономить память, заполняемую строками знаков, очень часто используемым знакам можно приписывать короткие коды и более длинные коды другим знакам. Этот метод в действительности дает слабую степень защиты, и сжатые сообщения можно затем перешифровать с использованием более сильного криптографического метода.

Словарные методы сжатия данных.

По существу, любой метод сжатия предполагает кодирование символов или групп символов с соответствующим бинарным кодом (код может быть не только бинарным).



В теории четного разделения между моделью преобразования и способа кодирования нет.



Моделью являются вероятностные свойства символов алфавита, а дальше кодирование основывается на замене символов с большей вероятностью появления на коды минимальной длины и наоборот, то есть замена на соответствующие бинарные последовательности или коды.

Вероятностные методы предполагают в общем случае передачу вместе со сжатыми данными определенного словаря. Пример наиболее простого словаря:

a_1 = 011
a_2 = 11
.
.
.
a_3 = 00001

Очевидно, если же на приемной или передающей стороне используется одна и та же модель, один и тот же метод кодирования, который заранее известен, то необходимости в передаче этого словаря вместе с данными нет. Этот словарь является статичным (неизменным) на каком-то промежутке времени.

Такие методы сжатия относятся к группе статичных методов или к группе с неизменяемым словарем. Этот метод можно назвать статическим статичным методом сжатия.

Примером такого метода сжатия являются архиваторы ZIP, RAR, ARJ.

В таблицах кода ASCII 1 символ представляется 1 байтом размером 8 бит. В случае использования вероятностного метода преобразования на основе статичного словаря, на основе таблицы ASCII, есть возможность вписать незадействованные 160 комбинаций в качестве кодов наиболее встречаемых бинарных или иных последовательностей.

В общем случае словарь может содержать произвольное число комбинаций произвольной длины. Основным вопросом является возможность составления словаря для всех комбинаций букв на основе алфавита русского языка. Следовательно, задача построения словаря — задача оптимизационная.

Рассмотрим пример:

Положим, что текст состоит из 4-ех буквенных слов на основе английского языка:

$$\underbrace{A\{a,b,c,d,\ldots,\underbrace{-,,,.,:,\ldots}_{6}\}}_{H=32} \qquad (1)$$

Если вероятность появления каждого знака в тексте одинакова, то каждый из этих знаков может быть зашифрован бинарным кодом длиной **5** бит. Полагаем, что вероятность появления каждого 4-х буквенного слова одинакова.

$32^4 = 2^{20} = 10048567$ — количество слов (возможно и 4-ех буквенных) (2)

Вероятность того, что данное слово будет равно какой-то конкретной комбинации:

$p = 1 / 10048567$

Если все комбинации разместить в словаре, то каждой комбинации должен соответствовать бинарный код длиной **20** бит. И таких слов у нас **10048567**.

Встает вопрос:

Что лучше **32** и **5** или **10048567** и **20**?

Оптимальное решение размера словаря находится между двумя этими крайними ситуациями (1) и (2).

Положим, что из миллиона слов все-таки имеются те, которые встречаются реже и чаще. В словарь размещаем 256 наиболее часто встречающихся слов, весь массив из 4-ех разрядных слов, общим объемом $\approx 2^{20}$, мы разделим на 2 подмассива:

- подмассив $\{X_4\}^1 = 2^8$ — в словаре.
- Подмассив $\{X_4\}^2 = 2^8$ — находящийся вне словаря.

Очевидно, в силу этого разделения, в силу выбранной модели, принцип кодирования должен быть разным.

Например:

Каждому из слов $\{X_4\}^1$ соответствует фиксированная бинарная последовательность из 8 бит, плюс 0, который дописывается впереди как флаг, что в сумме дает 9 бит, то есть в $\{X_4\}^1$:

$\{X_4\}^1 \rightarrow 0$ 8 бит $\rightarrow \sum = 9$ бит.

В подмассиве $\{X_4\}^2$:

$\{X_4\}^2 \rightarrow 1$ 20 бит $\rightarrow \sum = 21$ бит.

Интегральный коэффициент C соответствует среднему числу бит во всех кодовых комбинациях.

Если предположить, что вероятность равна p, то это означает, что слово есть в словаре из 256 единиц, если (p-1) — то этого слова в словаре нет. Тогда C = p*9 бит + (1-p)*21 бит = 9p + 21 – 21p = 21 –12p.

Стоит вопрос: каким должно быть p, чтобы размещение соответствующих слов в словаре было более эффективным в сравнении с методом размещения в словаре всех 2^{20} слов.

Если мы разместим все эти слова в словаре, то средняя длина C будет = 20, следовательно:

$21 - 12p < 20$.

$p > 1/12 \approx 0.084$.

При построении словаря исходят из следующих критериев:

1. скорость кодирования комбинаций и скорость обратного преобразования. Чем больше словарь, тем скорость ниже.
2. размеры требуемой памяти. Чем больше словарь, тем больше размер требуемой памяти.
3. эффективность сжатия. В первом приближении можно оценить параметром C. Чем меньше C, тем выше эффективность сжатия.

Наилучшим является подход, при котором словарь строится под тип данных. Важнейшей проблемой при проектировании словарной схемы является выбор размера кодового словаря. В связи с этим часто ограничение накладывается на размер фраз. Как правило, эти фразы имеют произвольную длину. Относительно размера фраз составление словаря может быть статичным полуадаптивным или адаптивным.

В теории метод разбиения текста на фразы для их дальнейшего кодирования называется разбором текста.

Разбор текста обычно делят на:

- тщательный;
- оптимальный.

Рассмотрим пример:

Пусть словарь W состоит из следующих слов:

W: a, b, c, aa, aaaa, ab, baa, bccb, bcccba.

Предполагают, что используется 4-ех символьный алфавит. Очевидно, что в словаре имеется 9 слов, различной длины. В словаре 9 комбинаций, каждая комбинации соответствует код одинаковой длины 4 бита.

Имеется строка символов или фраза текста: $\mathbf{X_k}$ **= aaabccbaaaa**, которую нужно сжать. Полуадаптивный алгоритм предполагает создание словаря под конкретный тип документов. Адаптивный — строит словарь под каждый

документ. Считаем, что рассматриваемый параметр соответствует полуадаптивному методу, то есть словарь составляется под группу документов. Разбор данной строки определяет набор [bcccba] как самый длинный.

Многие алгоритмы основаны на поиске самой длинной из последовательностей. Проблема здесь в том, какой должна быть длина максимальной последовательности, и на каком интервале искать эту последовательность. Можно разобрать строку так:

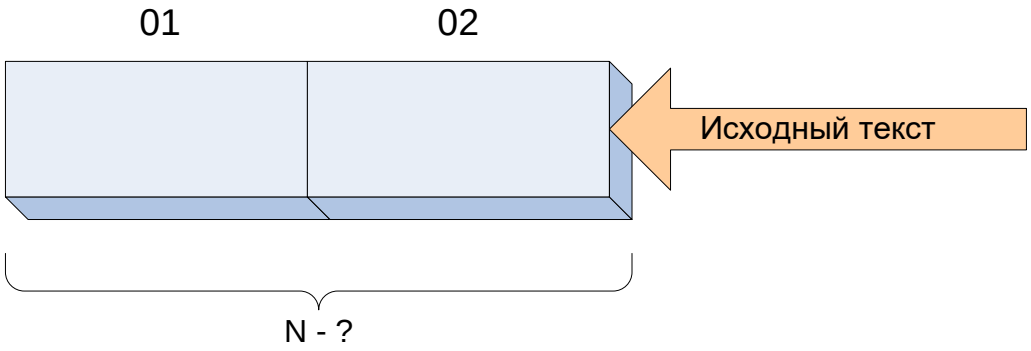
a aa bccba aa a — общая длина 4*5 = 20 бит, а можно и так:

aa a bccb aaaa — общая длина 4*4 = 16.

До появления метода, описанного Лемпелем-Зивом, все методы были в основном статическими, то есть модель основывалась на априорном анализе, словарь был статическим. В случае полуадаптивного метода, как правило, вместе с данными требовалось пересылать словарь.

С появлением методов Лемпеля-Зива, было положено начало использования адаптивных методов сжатия, где не нужно было передавать словарь.

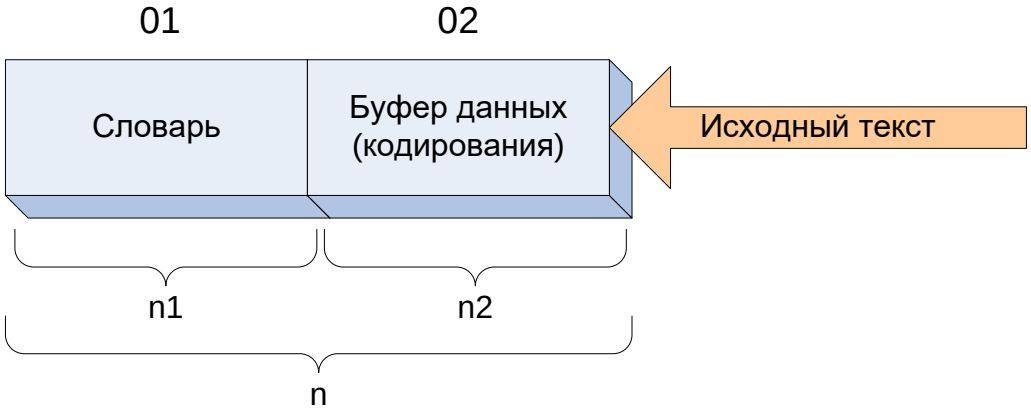
Основной метод состоит в поиске комбинаций, которые уже встречались, и замене этого повтора на комбинации символов **i**, **j** (**p**, **q**), где **i** — индекс начала комбинации в передаваемом тексте, а **j** — длина этой повторяющейся комбинации. Следовательно, в методе Лемпеля-Зива существуют как бы 2 окна или 2 буфера.



Как правило, N — это сотни символов. Окно 02 — буфер данных, в него “въезжают” данные (исходный текст), которые нужно сжимать, кодировать. Далее, эти данные “въезжают” в 01 окно — словарь, в нем ищутся комбинации, повторяющиеся в фрагменте 02.

Реализация алгоритма Лемпеля-Зива.

Преобразование осуществляется по принципу размещения и передачи исходного сообщения через буфер, который делится на 2 части: буфер данных (буфер кодирования) и словарь. Если общая длина **n**, то:

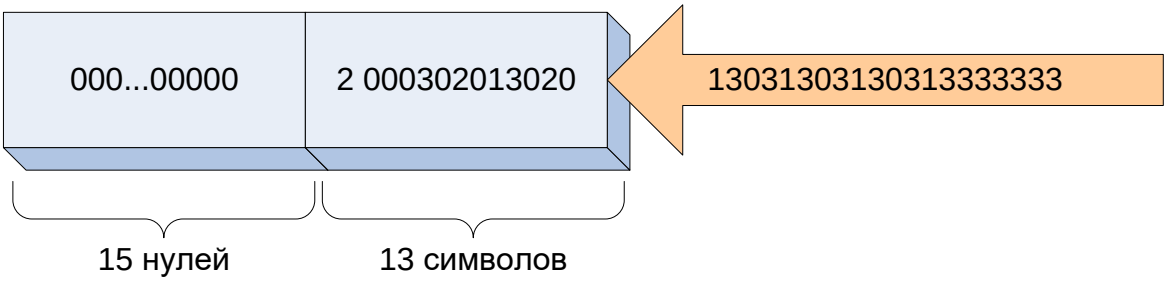


$n = n_1 + n_2$ символов или разрядов.

Принцип состоит в том, чтобы в тексте находить повторяющиеся комбинации, причем анализируемые данные находятся в буфере кодирования, а повторение должно быть найдено в словаре. Основной принцип: найденный повторяющийся ряд символов в буфере данных заменяется, в основном, парой (**p**, **q**) символов, где **p** — индекс символов в словаре, являющийся началом повтора в буфере данных, индекс от 1 до n_2. Число **q** — целое число, обозначающее длину повторяющейся комбинации, и соответствующее такой же комбинации в словаре. Кроме этого добавляется еще один символ $\mathbf{X_i}$, который является частью потока $\mathbf{X_k}$ и следует за найденным повторением в буфере данных. Буфер работает по принципу сдвига регистра. Пара (**p**, **q**) — диада символов, заменяющая повторения.

Рассмотрим пример:
Используем для передачи данных алфавит, состоящий из знаков:
A{0, 1, 2, 3}, M = 4.
$\mathbf{X_k}$ **= 2000302013020130313031303133333333**
n = 28.
n_1 (буфер данных) = 13 символов.
n_2 (длина словаря) = 15 символов.
В четверичном представлении p и q:
$0_4 = 00$, $1_4 = 01$, $2_4 = 02$, $3_4 = 03$, $4_4 = 10$, $5_4 = 11$.

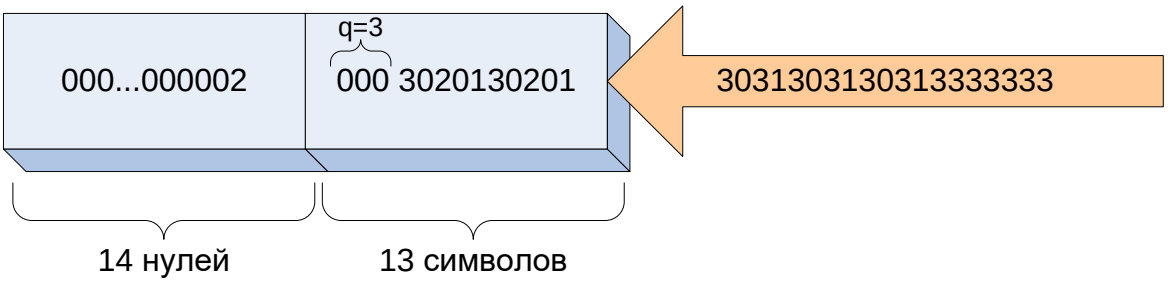
1. на первом шаге:



Анализируем 1-й символ в буфере кодирования на предмет соответствия (наличия) такого же символа или нескольких символов в словаре. Таких символов нет, следовательно, **p = 0**. Длина повторения **q = 0**. Таким образом, имеем следующую триаду:

(p, q, X_i) = (**00 00 2**).

2. на втором шаге:



Находим повторение (**000**), длина этого повторения **q = 3**. Поскольку в словаре нулевые символы записываются с 1-ой по 14-ю позицию, то индексом p — началом повторения, может быть выбрано любое число от 1 до 12:

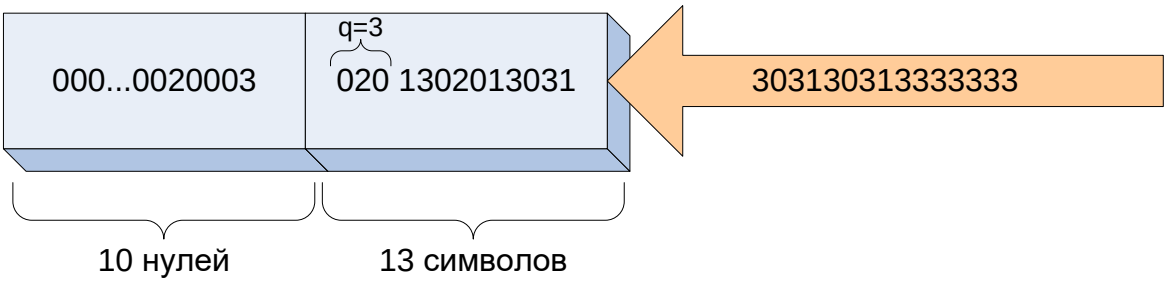
$1 \le p \le 12$;
$p = 6$.

Итак, получим следующую триаду:

(p, q, X_i) = (**6, 3, 3**) = (**12 03 3**).

Передвигаем на q + 1 = 3+1 = 4.

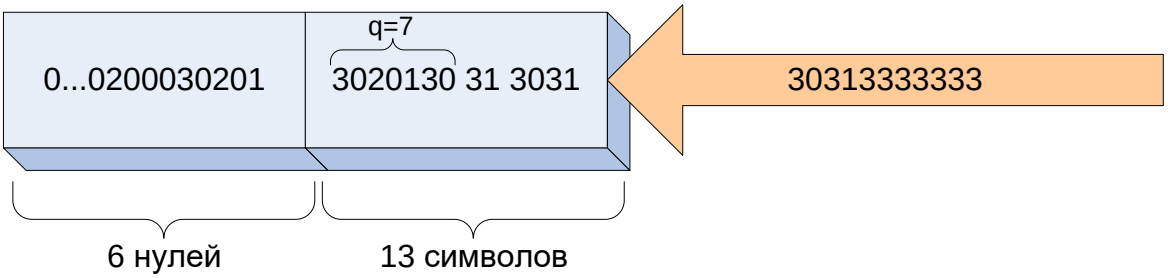
3. на третьем шаге:



В этой ситуации наиболее длинный повтор **020**, **q = 3**, **p = 10**.
Передвигаем на q + 1= 4. И получаем триаду:

(p, q, X_i) = (**10, 3, 1**) = (**22 03 1**).

4. на четвертом шаге:



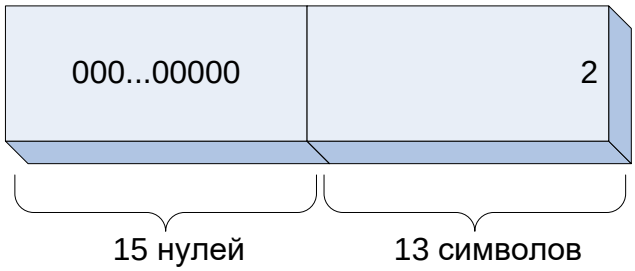
В этой ситуации наиболее длинный повтор **3020130**, **q = 7**, **p = 11**.
Передвигаем на q + 1= 8. И получаем триаду:

(p, q, X_i) = (**11, 7, 3**) = (**23 13 3**).

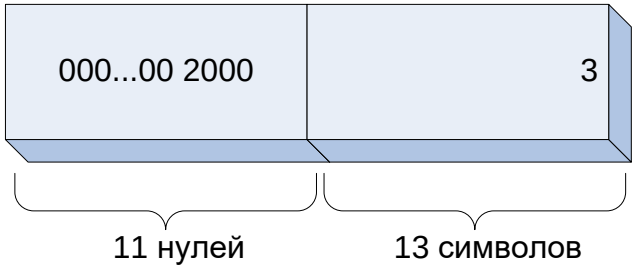
Задача обратного преобразования:

Имеем в 4-ой системе такие данные: 00002 12033 22031 23133 30301 02013 32103.

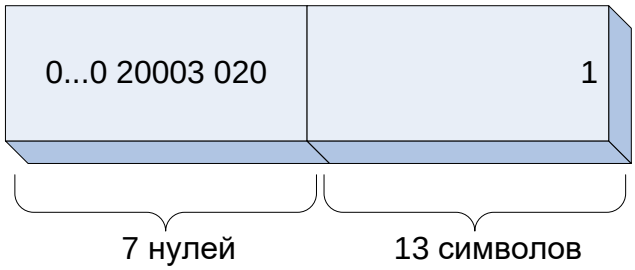
В исходном состоянии в окно словаря записывают 15 нулей.



Дальше анализируем следующие 5 символов: 12033, следовательно **q = 3** и **p = 6**. Поскольку в словаре содержаться пока только одни нули, то мы можем сделать вывод, что наш повтор это 000. В этот буфер вписывается 000 справа, а в буфер кодирования 3.



Дальше анализируем следующие 5 символов: 22031, следовательно **q = 3** и **p = 10**. Отсчитываем десять 0 и получаем начало следующей последовательности: 020, записываем ее в буфер, следующим символом будет 1, ее записываем в буфер кодирования.



И так далее.

Хронология версий метода Лемпеля-Зива:

1977 г. — 1-я версия. Эта версия предполагает, что указатели (p, q) и символы (X_i) чередуются. Указатели адресуют подстроку к предыдущим n символам.

1978 г. — версия L-Z 78. Указатели и символы так же чередуются, но указатели адресуют ранее разобранную строку.

1981 г. — версия L-Z-R (Roden). Указатели и символы чередуются, указатели адресуют подстроку среди всех предыдущих символов.

1984 г. — версия L-Z-W (Welch). Выходная информация содержит только указатели, которые имеют фиксированную длину и адресуют ранее проанализированную строку.

1985 г. — версия L-Z-C (Thomas). Вводятся ограничения на длину указателей.

1986 г. — версия L-Z-SS (Bell). Указатели и символы разделены флажком.

Принципы построения компрессоров с потерей информации.

Компрессоры обрабатывают мультимедийную информацию. Основной принцип работы базируется на слуховых, психоэмоциональных и зрительных особенностях человека.

Криптографические методы преобразования информации.

Основная особенность криптографического метода преобразования информации:

$\{X_k\} \neq \{Y_n\}$;
$k = n$

За исключением методов генерации электронно-цифровой подписи (ЭЦП).

Главная цель методов: повышение уровня конфиденциальности (тайности) передаваемой информации.

Проблемой защиты информации путем ее преобразования занимается криптология (*krypt*o*s* - тайный, l*ogos* - наука). Криптология разделяется на два

направления - криптографию и криптоанализ. Цели этих направлений прямо противоположны.

Криптография занимается поиском и исследованием математических методов преобразования информации.

Сфера интересов криптоанализа — исследование возможности расшифровывания информации без знания ключей.

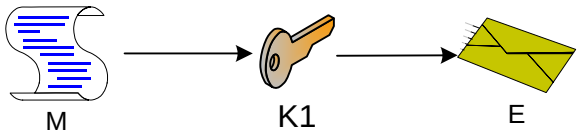
Криптология

Криптография Криптоанализ

Базовые понятия и определения из области криптографии:

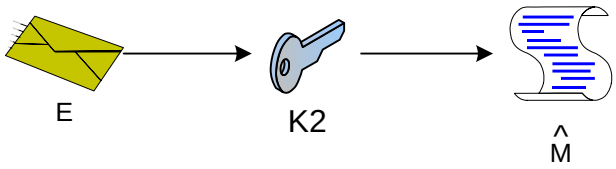
1. Сообщения, подлежащие зашифровке, являются сообщения или текстом открытым (явным).
2. Зашифровка — это операция или процедура превращения сообщения открытого в сообщение закрытое (тайное или зашифрованное). Зашифрованное сообщение называется шифрограммой. Процедура или алгоритм зашифровки опирается на конкретные методы.
3. Обратная процедура, то есть превращение зашифрованного сообщения в исходное сообщение называется расшифрованием.
4. Процедура или алгоритм криптопреобразований опирается на использование так называемых ключей.
5. Ключ — информация, с помощью которой преобразовывается исходное сообщение в шифрограмму и наоборот.

Схематично, общий вид криптопреобразования можно представить:



M — Message, сообщение.
K_1 — Key, ключ.
E — Encryption, зашифрованное сообщение.

Обратная процедура:



В зависимости от того, какие ключи используются и как различаются методы и алгоритмы криптопреобразования.

Классификация методов криптопреобразования.

1. Первый классификационный признак основан на типе ключа. Все системы делятся соответственно на две большие группы:

- Системы с ключом тайным, когда ключ известен только отправителю и получателю сообщения, и на каждой из сторон используется тот же самый ключ, то есть K_1 и K_2. Такие системы называются симметричными системами криптопреобразования.
- Может быть ситуация, когда ключ K_1 не равен в точности K_2 , такие системы называются с ключом “публичным” или асимметричными системами криптопреобразования.

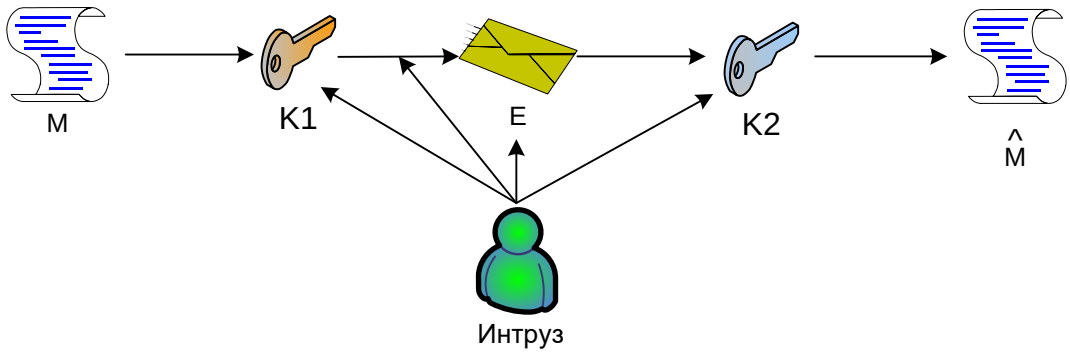
2. Второй классификационный признак касается формирования шифрограммы на основе сообщения M:

- Блочное шифрование, когда все сообщение произвольной длины делится на блоки и отдельно зашифровывается каждый из блоков тем же самым ключом. Тот же самый метод, что и в помехоустойчивом блочном кодировании.
- Поточное шифрование. По существу, создаются псевдобесконечная шифрограмма. То есть преобразованию подвергается целый поток данных. Шифрование зависит от всех проанализированных до этого символов.

3. Третий классификационный признак касается позиции преобразованных символов в шифрограмме относительно исходных данных. Здесь все шифры делятся на:

- Подстановочные.
- Перестановочные.

Обобщенная схема:



Интруз — физическое лицо или процесс, которое реализует неразрешенный или несанкционированный доступ к информации закрытой ее владельцем.

Если владелец информации закрыл доступ к M, то действие интруза направлено на получение K_1 и K_2 ключей или зашифрованного текста E.

Одним из способов осуществления расшифровки зашифрованного текста интрузом может быть атака “в лоб”, то есть перебор всех возможных ключей. У интруза существует несколько целей:

1. получить ключ;
2. получить сообщение и прочитать;
3. подменить зашифрованное сообщение, зная ключ.

Выводы:

Второй целью (помимо закрытия сообщения от несанкционированного доступа) является возможность идентификации отправителя сообщения и возможной аутентификации сообщения или установления подлинности отправителя сообщения.

Установление подлинности может основываться на использовании информации о ключе, либо на использовании более современного метода, который называется электронная цифровая подпись.

Базовые методы шифрования

Сущность подстановочного шифрования состоит в том, что, как правило, исходный текст (M) или зашифрованный текст (M) используют один и тот же

алфавит, а ключом является подстановка вместо одних символов других символов.

Примером такого шифрования является шифр Цезаря:

O: {a, b, c, d,…}.

K: {c, d, e, f…}, который получается путем сдвига на 3 символа алфавита включительно, то есть k=3.

Если исходное сообщение M (X_k) = “BCDF”, то зашифрованное сообщение E = “DEFH”.

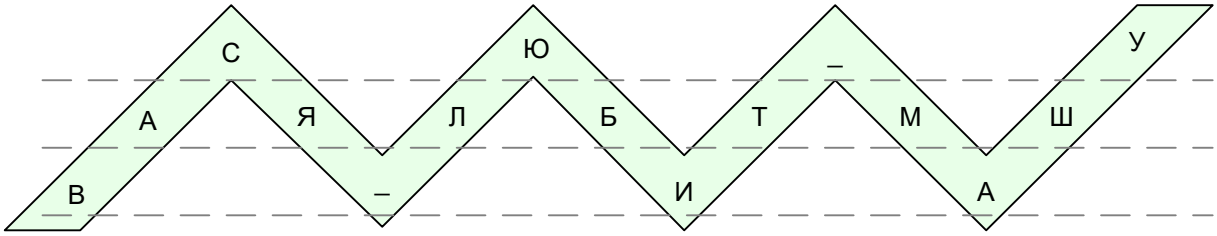
Ключ **k** может быть произвольным, алфавит тоже может быть другим. Может, например, быть:

A {A, B, C, D}

K {*, ↓, ↑, 1}

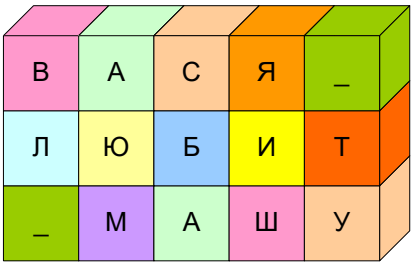
Перестановочные шифры используют перестановку символов исходного сообщения в соответствии с установленным правилом.

Пример 1: M= ‘ВАСЯ ЛЮБИТ МАШУ’.



E= СБА…АЯЮИМШ…В_ЛТ_У

Пример 2: M= ‘ВАСЯ ЛЮБИТ МАШУ’.



E= В_ИА…АЛТШ…СЮ_У…ЯБМ

Если сообщение состоит из M символов, то зная алфавит A{a_i}, $i=\overline{1,N}$, то перестановок будет $\mathbf{N^k}$.

Все современные шифры являются подстановочно-перестановочными. Для того чтобы затруднить процедуру поиска шифров, используются так называемые

односторонние функции преобразования, или так называемые “математические аспекты шифрования”.

Шифрование, основанное на проблеме “дискретного логарифма”. Если есть числа a, x, n, такие, что можно вычислить:

$b = a^x \bmod n$.

Однако, если известны **b**, **a** и **n**, то **x** можно найти на основе вычисления логарифма. Это единственный путь поиска $\mathbf{X_i}$. Оказывается, что при определенных значениях **b**, **a** и **n**, найти **X** является большой проблемой, а так как найти **X,** значит найти ключ, то это и есть проблема дискретного логарифма.

Зная **f(x)** и **X**, легко найти **Y**, но, зная **f(x)** и **Y**, весьма проблематично найти **X** — это и есть использование однонаправленной функции.

Использование эллиптических кривых. Выбор точек на этой кривой (взаимосвязанных точек) является ключом, зная кривую и даже зная одну точку, чрезвычайно трудно найти другую или другие взаимосвязанные точки.

Симметричные или асимметричные системы, как правило, базируется на проблеме поиска **X**, причем значения **a**, **x** и **n** — это очень большие числа (целые, положительные). Современные системы имеют порядок или длину больше 1000 двоичных символов.

Характеристика симметричных систем:

В симметричных системах Отправитель и Получатель используют один и тот же ключ, который должен быть известен только им.

Рассмотрим пример:

Сообщение $\mathbf{M_{(0,1)}}$**= 10101100**

$K_1 = K_2 = 1010$. Используется самая простая операция шифрования:

$\boldsymbol{E_i = M_i \oplus K}$;

$\boldsymbol{\bar{M}_i = E_i + K}$, где i — i-тый блок шифрования, M_i — i-тая часть сообщения.

$$E_1 = \begin{array}{r} 1010 \\ \oplus\ 1010 \\ \hline 0000 \end{array}$$

$$E_2 = \begin{array}{r} 1100 \\ \oplus\ 1010 \\ \hline 0110 \end{array}$$

$$E = E_1 + E_2 = 00000110$$

$$\hat{M}_1 = \begin{array}{r} 0000 \\ \oplus\ 1010 \\ \hline 1010 \end{array}$$

$$\hat{M}_2 = \begin{array}{r} 0110 \\ \oplus\ 1010 \\ \hline 1100 \end{array}$$

$$\hat{M} = 10101100 = M$$

$\oplus$ — сумма по модулю 2.

Современные системы используют множественные арифметическо-логические преобразования исходного текста.

Первый из стандартов, реализующий данный метод был DES (Data Encryption Standard).

DES (Data Encryption Standard)

Стандарт шифрования DES (Data Encryption Standard) был разработан в 1970-х годах, он базируется на алгоритме DEA.

Исходные идеи алгоритма шифрования данных DEA (data encryption algorithm) были предложены компанией IBM еще в 1960-х годах и базировались на идеях, описанных Клодом Шенноном в 1940-х годах. Первоначально, эта методика шифрования называлась Lucifer, разработчик Хорст Фейштель, название dea она получила лишь в 1976 году. Lucifer был первым блочным алгоритмом шифрования, он использовал блоки размером 128 бит и 128-битовый ключ. По существу этот алгоритм являлся прототипом DEA. В 1986 в Японии (NIT) разработан алгоритм FEAL(Fast data Encipherment ALgorithm), предназначенный для использования в факсах, модемах и телефонах (длина ключа до 128 бит). Существует и ряд других разработок.

DEA оперирует с блоками данных размером 64 бита и использует ключ длиной 56 бит. Такая длина ключа соответствует 10^{17} комбинаций, что обеспечивало до недавнего времени достаточный уровень безопасности. В дальнейшем можно ожидать расширения ключа до 64 бит (например, LOKI) или вообще замены DES другим стандартом.

Входной блок данных, состоящий из 64 бит, преобразуется в выходной блок идентичной длины. Ключ шифрования должен быть известен, как

отправляющей так и принимающей сторонам. В алгоритме широко используются перестановки битов текста.

Вводится функция f, которая работает с 32-разрядными словами исходного текста (A) и использует в качестве параметра 48-разрядный ключ (J). Схеме работы функции f показана на рис. 6.4.1.1. Сначала 32 входные разряда расширяются до 48, при этом некоторые разряды повторяются. Схема этого расширения показана ниже (номера соответствуют номерам бит исходного 32-разрядного кода).

2

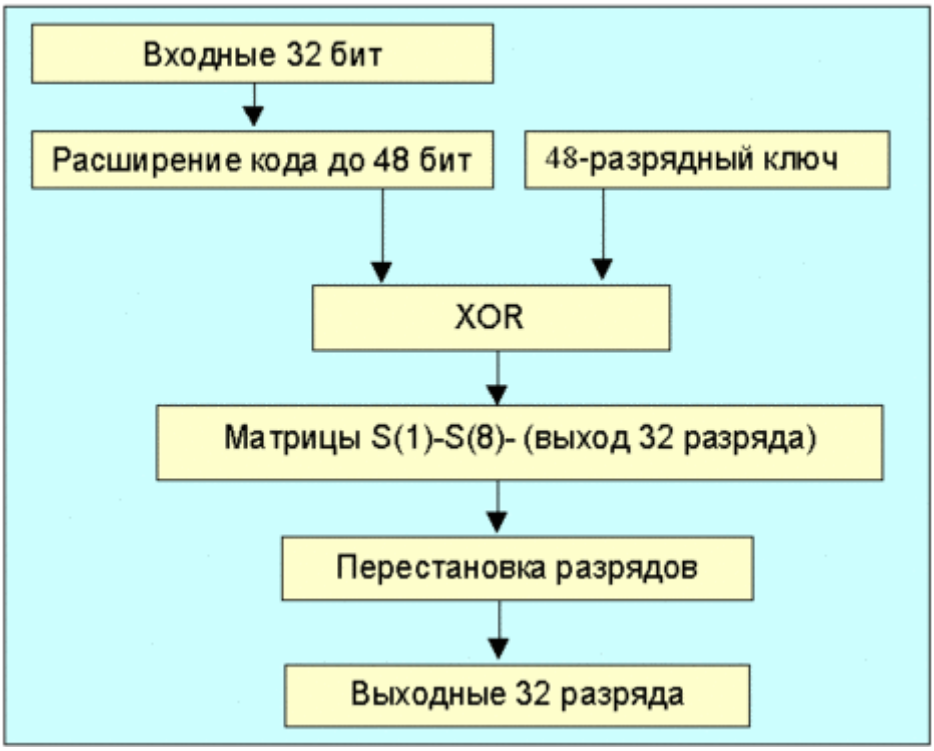
		0	1	2	3
2	3	4	5	6	7
6	7	8	9	0	1
0	1	2	3	4	5
4	5	6	7	8	9
8	9	0	1	2	

Для полученного 48-разрядного кода и ключа выполняется операция исключающее ИЛИ (XOR). Результирующий 48-разрядный код преобразуется в 32-разрядный с помощью S-матриц. На выходе S-матриц осуществляется перестановка согласно схеме показанной ниже (числа представляют собой порядковые номера бит).

6		0	1
9	2	8	7
	5	3	6
	8	1	0
		4	4
2	7		

9 3 0

2 1 5



Алгоритм работы функции f

Преобразование начинается с перестановки бит (**IP – Initial Permutation)** в 64-разрядном блоке исходных данных. 58-ой бит становится первым, 50-ый – вторым и т.д. Схема перестановки битов показана ниже.

8 0 2 4 6 8 0

0 2 4 6 8 0 2

2 4 6 8 0 2 4

4 6 8 0 2 4 6

7 9 1 3 5 7

9 1 3 5 7 9 1

1 3 5 7 9 1 3

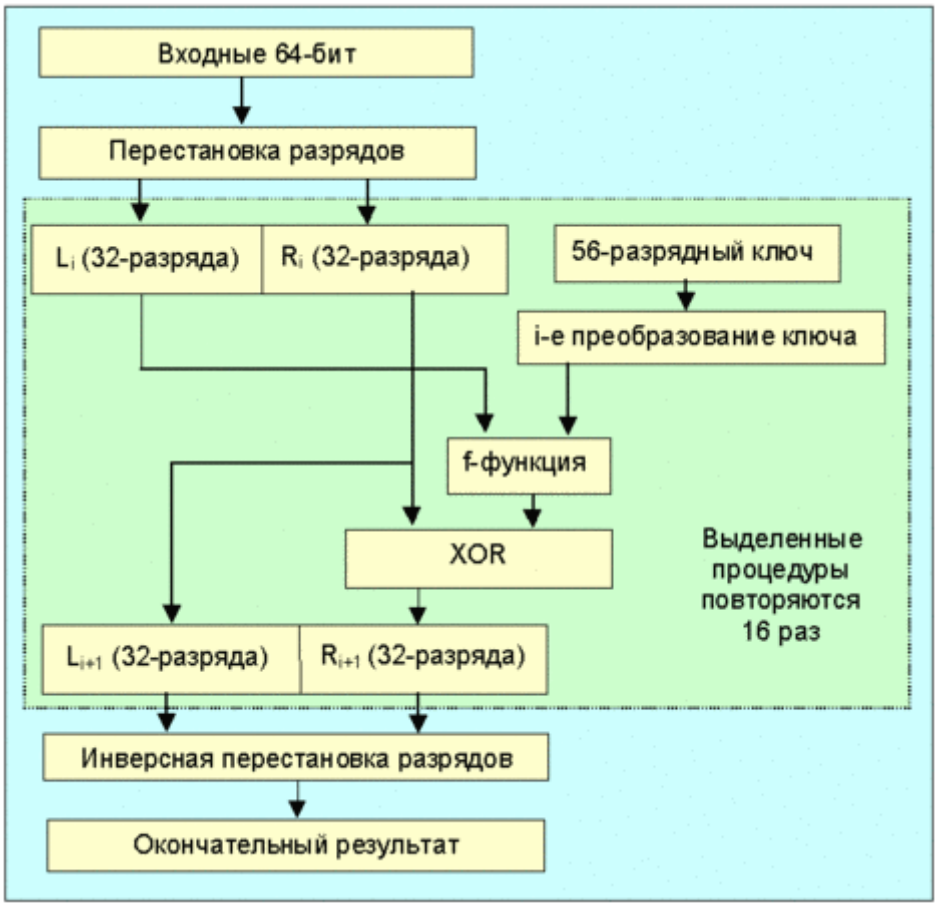
3 5 7 9 1 3 5

Полученный блок делится на две 32-разрядные части L_0 и R_0. Далее 16 раз повторяются следующие 4 процедуры:

- Преобразование ключа с учетом номера итерации i (перестановки разрядов с удалением 8 бит, в результате получается 48-разрядный ключ)

Пусть $A=L_i$, а J – преобразованный ключ. С помощью функции f(A,J) генерируется 32-разрядный выходной код. Выполняется операция XOR для R_i f(A,J), результат обозначается R_{i+1}. Выполняется операция присвоения $L_{i+1}=R_i$.

Структурная схема реализации алгоритма dea показана на следующем рисунке.



Структурная схема реализации алгоритма DEA

Инверсная перестановка разрядов предполагает следующее размещение 64 бит зашифрованных данных (первым битом становится 40-ой, вторым 8-ой и т.д.).

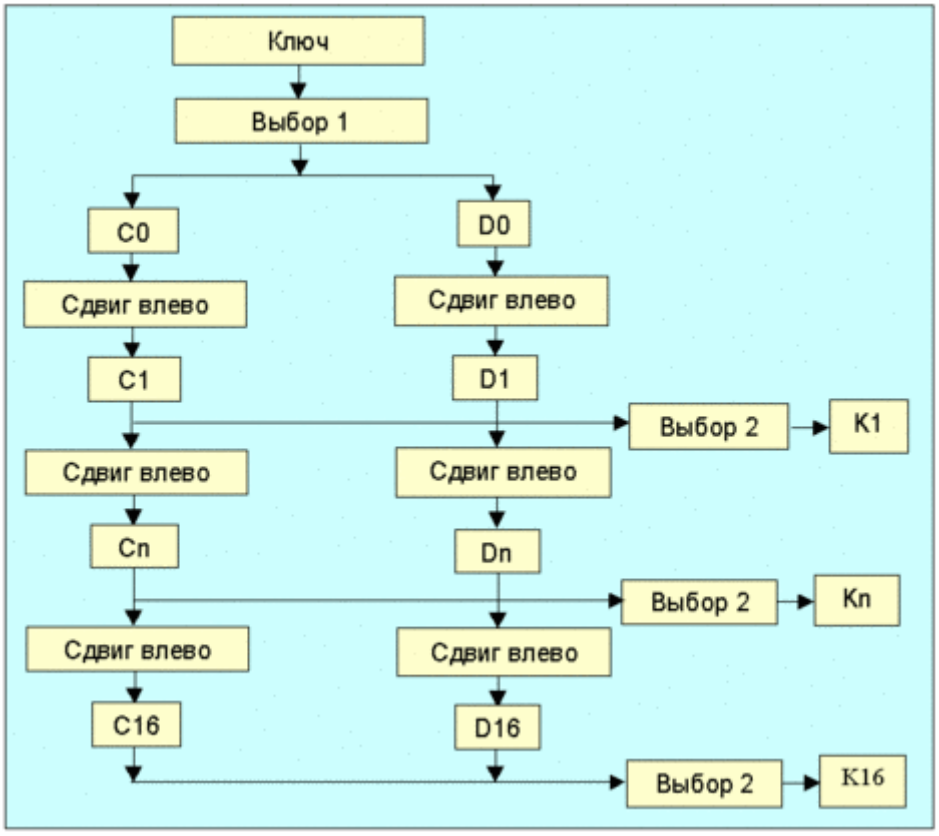
0	8	6	6	4	4	2
9	7	5	5	3	3	1
8	6	4	4	2	2	0
7	5	3	3	1	1	9
6	4	2	2	0	0	8
5	3	1	1	9	9	7
4	2	0	0	8	8	6
3	1		9	7	7	5

S-матрицы представляют собой таблицы содержащие 4-ряда и 16 столбцов. Матрица S(1) представлена ниже (эта матрица, также как и те, что приведены выше, являются рекомендуемыми).

											0	1	2	3	4	5
	4		3			5	1			0		2				
		5			4		3		0		2	1				
			4		3			1	5	2				0		
	5	2								1		4	0			3

Исходный 48-разрядный код делится на 8 групп по 6 разрядов. Первый и последний разряд в группе используется в качестве адреса строки, а средние 4 разряда – в качестве адреса столбца. В результате каждые 6 бит кода преобразуются в 4 бита, а весь 48-разрядный код в 32-разрядный (для этого нужно 8 S-матриц). Существуют разработки, позволяющие выполнять шифрование в рамках стандарта DES аппаратным образом, что обеспечивает довольно высокое быстродействие.

Преобразования ключей K_n (n=1,…,16; K_n = KS(n,key), где n – номер итерации) осуществляются согласно следующему алгоритму.



Алгоритм вычисления последовательности ключей K_n

Для описания алгоритма вычисления ключей K_n (функция KS) достаточно определить структуру “Выбора 1” и “Выбора 2”, а также задать схему сдвигов влево (табл. 6.4.1.2). “Выбора 1” и “Выбора 2” представляют собой перестановки битов ключа (PC-1 и PC-2; табл. 6.4.1.1). При необходимости биты 8, 16,…, 64 могут использоваться для контроля четности.

Для вычисления очередного значения ключа таблица делится на две части C0 и D0. В C0 войдут биты 57, 49, 41,…, 44 и 36, а в D0 – 63, 55, 47,…, 12 и 4. Так как схема сдвигов задана (табл. 6.4.1.2) C1,D1; Cn, Dn и так далее могут быть легко получены из C0 и D0. Так, например, C3 и D3 получаются из C2 и D2 циклическим сдвигом влево на 2 разряда

Таблица 1

PC-1 (Выбор 1)	**PC-2 (Выбор 2)**
57 49 41 33 25 17 9	14 17 11 24 1 5

1 58 50 42 34 26 18	3 28 15 6 21 10
10 2 59 51 43 35 27	23 19 12 4 26 8
19 11 3 60 52 44 36	16 7 27 20 13 2
63 55 47 39 31 23 15	41 52 31 37 47 55
7 62 54 46 38 30 22	30 40 51 45 33 48
14 6 61 53 45 37 29	44 49 39 56 34 53
21 13 5 28 20 12 4	46 42 50 36 29 32

Таблица 2

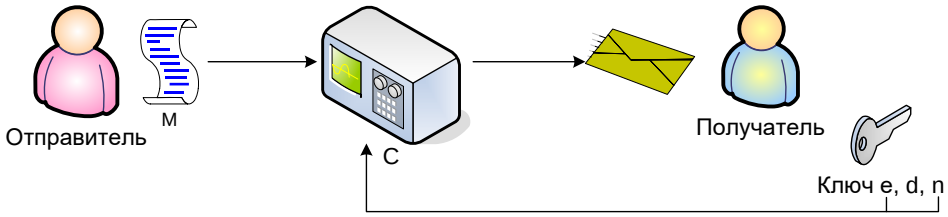
Номер итерации	**Число сдвигов влево**
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Криптографические системы с открытым (публичным) ключом. Ассиметричное шифрование.

Математическое решение проблемы перехода на ассиметричное шифрование нашли Диффи (Diffy) и Хеллман (Hellmann). Ключ состоит из тройки больших целых чисел **e, d, n**. Числа e и d — являются взаимно простыми (нет общего делителя кроме 1). Эти числа выбираются на основе определенной последовательности друг с другом и на основе зависимости от n. Числа **e**, **d** и **n** являются общедоступными и составляют одну часть ключа, которая называется открытой. Эти ключи хранятся в специальных сертификационных центрах. Задачей этих центров является подтверждение аутентичности этого ключа.

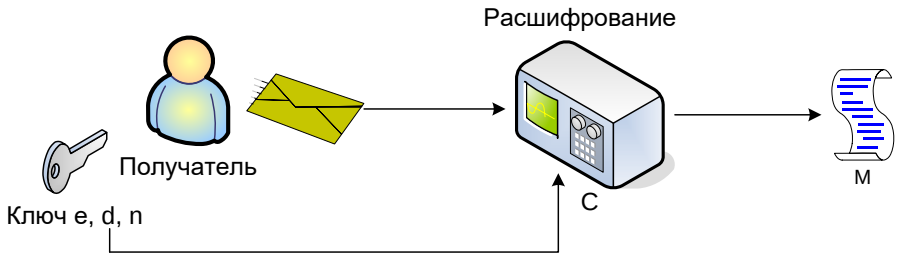
Число **d** является тайным и является известным только обладателям общего ключа.

Взаимозависимость между тремя этими параметрами основывается на однонаправленной функции.



M — сообщение, которое отправляет Отправитель, зашифровывается с помощью e и n. C — шифрование. Однако обратная операция получения M, не возможна:

А вот дешифрование может провести только Получатель:



Электронная цифровая подпись (ЭЦП).

Выполняет следующие функции:

1. удостоверяет
2. подтверждает
3. идентифицирует

Рассмотрим эти пункты поподробнее:

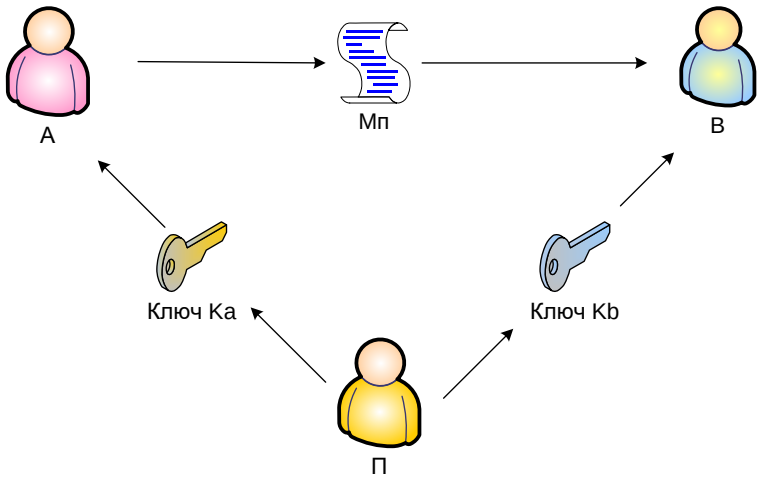
1. Подпись должна быть достоверна, это означает, что подписавший документ человек сделал это осознано, без принуждения.
2. Подпись неподдельна. Подписавший — автор подписи.
3. Подпись невозможно использовать повторно, мошенник не должен иметь возможность переносить подпись без ведома подписавшегося.
4. Подписанный документ не может быть изменен, особенно, если сделано несколько копий.
5. От подписи нельзя отречься.

Роль подписи такая же, как и в бумажных документах, но есть отличия. Нет проблем изменить любой файл. Подпись можно отсканировать и разместить на документе.

Алгоритмы и методы используемые ЭЦП:

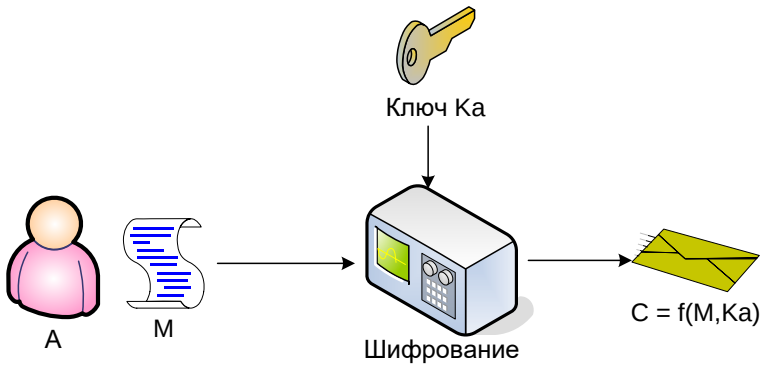
1. На основе симметричных криптосистем и посредника.

Пусть имеется абонент А, который пересылает сообщение абоненту В. Основная проблема — в механизме передачи и механизме хранения. Схематично рассмотрим схему с посредником:

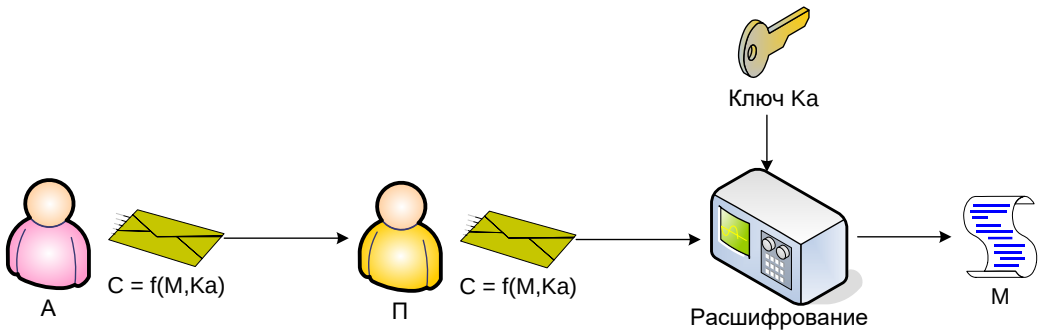


П — посредник, абсолютно доверенное лицо. Доверенное лицо посылает абоненту А ключ K_a и он же посылает абоненту В ключ K_b.

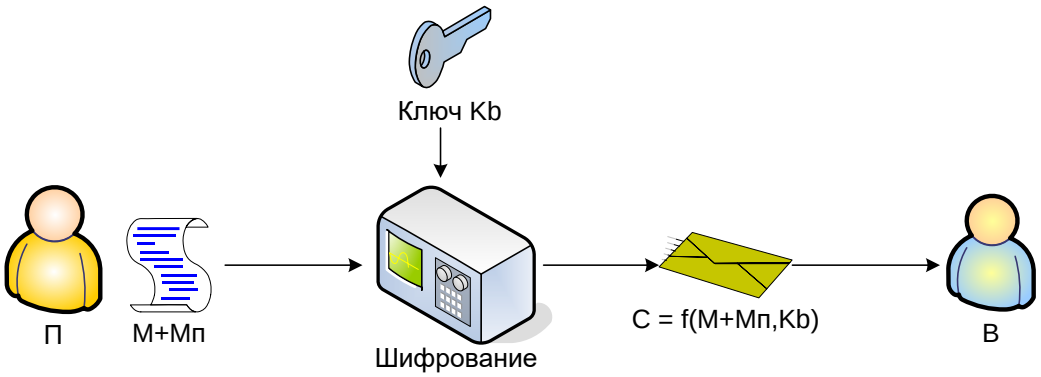
1-ый шаг: абонент А шифрует с помощью ключа K_a сообщение М и на основе сообщения М и ключа K_a получается шифрограмма С:



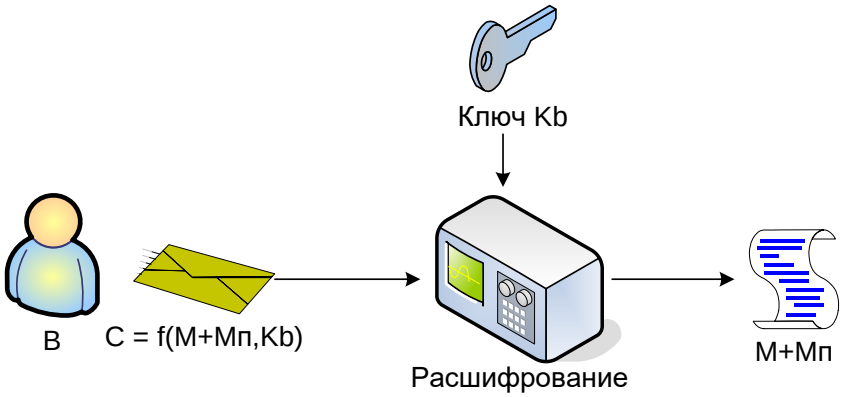
2-ый шаг: абонент А отсылает зашифрованное сообщение посреднику. Посредник расшифровывает сообщение с помощью ключа K_a получает сообщение М, что является подтверждением того, что документ прислан именно абонентом А:



3-ий шаг: посредник передает сообщение абоненту В с добавлением: “Да, это сообщение действительно от абонента А”. И шифрует это сообщение:



4-ый шаг: пользователь В получает шифрограмму С и с помощью ключа K_b расшифровывает сообщение с подтверждением от посредника об подлинности авторства А:



Выводы:

Подпись действительно достоверна, поскольку посредник является абсолютно достоверным лицом.

Подпись неподдельна. Если кто-то попытается выдать себя за А, то это должен выдать посредник, при условии, что абонент А соблюдает конфиденциальность своего ключа.

Подпись невозможно использовать повторно. Если, к примеру, D захочет это сделать, то лицо А и лицо В это могут опровергнуть.

Невозможно изменить подпись, с учетом того, что это можно легко обнаружить.

Подпись невозможно отрицать. Вступает принцип мажоритарности.

Основная проблема:

Чем больше пользователей, тем труднее реализовывать посреднику свои функции. Если посредником является компьютер, то эта проблема отсутствует, зато появляется другая: что если он перестанет работать?

Для разрешения этой проблемы был разработан следующий метод.

2. На основе криптосистем с открытым ключом (RSA).

Р. Риверст разработал алгоритм и метод, используемый ЭЦП на основе данного алгоритма. В данном методе А и В имеют сгенерированные личные ключи **{E_a, D_a, M_a; E_b, D_b, M_b}**. Между классическим шифрованием по методу асимметричных систем и ЭЦП существует разница.

Между формированием ЭЦП на основе криптосистем с открытым ключом и простым шифрованием на основе того же метода имеется одна существенная особенность. Если используется обычное шифрование между А и В, где А пересылает сообщение В, то А использует для шифрования открытый ключ В, для расшифрование В использует свой тайный ключ, то есть использует ключи получателя. В системах с ЭЦП на основе криптосистем с открытым ключом, если А отправляет подписанное сообщение В, то для подписи документа А использует свой собственный тайный ключ для расшифрования. Таким образом, для получения подтверждения авторства документа, получатель использует открытый ключ отправителя, то есть использует в данном случае ключи, принадлежащие отправителю.

Протокол генерации использования ЭЦП на основе RSA.

1. Абоненты А и В независимо, либо с помощью специализированного центра генерирует пары простых: **p_a, q_a** и **p_b, q_b**. С помощью них вычисляются n_a и n_b;
2. **$n_a = p_a \cdot q_a$** и **$n_b = p_b \cdot q_b$**;
3. Вычисляются ключи: **{e_a, d_a ; e_b, d_b}**.

Числа **{n_a, e_a; n_b, e_b}** помещаются в общедоступный центр и получают статус открытых, публичных ключей.

Для обмена сообщениями, а именно для формирования цифровой подписи, каждая из сторон использует ключ абонента, подписавшего сообщение, а именно: отправитель использует собственный тайный ключ, а получатель — ключ отправителя.

Как и в системах с симметричным ключом, может быть реализован следующий алгоритм:

Отправитель для шифрования использует публичный ключ получателя, для расшифрования получатель использует свой тайный ключ. Однако, в этом случае трудно доказать или оспорить, кто является подписантом (подписавшим документ) документа.

Разработан третий и основной метод: ЭЦП на основе асимметричных криптосистем и однонаправленных хэш-функций.

3. ЭЦП на основе асимметричных криптосистем и однонаправленных хэш-функций.

В математике и информатике есть такие функции, которые принимают на входе строку данных произвольной длины и преобразовывают ее в строку меньшей длины. Они называются значением хэш-функции или дайджестом (Digest). Функция, которая используется для преобразования, должна быть односторонняя.

Длина сообщения H, **k** — длина входной последовательности, **k`** — выходной последовательности. Значение k` никак не зависит от k, однако, поскольку **k`** << **k**, то одной и той же функции могут соответствовать различные входные комбинации, которые на выходные комбинации не оказывают влияния. Если в сообщении M изменится хотя бы один бит, то в хороших значениях хеш-функции изменится около половины всех символов.

Характеристика хэш-функции:

Общий метод вычислений хэш-функций. Как правило, хэш-функция является конкатенацией различных составных частей или блоков, то есть хэш-функция есть подсоединение блоков:

$$\mathbf{H=h_1 + h_2 + \ldots + h_i.}$$

В современных системах i меняется от 3 до 6(7), причем вычисление каждого последующего блока хэш-функции зависит не только от данных, но и от значений предыдущих блоков.

$\mathbf{h_j= F(M, h_{j-1})}$, где j меняется от 2 до i.

На практике однонаправленность функции контролируется на основе принципа сжатия (сжимающей функции). Принцип сжимаемости функции означает, что рассматривается данное не целиком, а по частям.

При создании значений хеш, все сообщение делится на блоки. Принцип разделения на блоки различен при различных алгоритмах.

Существует MD (Message Digest)-серия: MD2, MD4, MD5.

Рассмотрим функции MD4:

Функция MD4 разработана Роном Райвестом (Ron Rivest). Функция возвращает значение = 128 разрядам, то есть **k`** (MD4) = 128 бит, вне зависимости от длины сообщения. Хэш-функция всегда представляется в двоичной форме. Как формируются эти 128 бит? Сообщение **M** должно претерпеть предварительную обработку, для того, чтобы длина входной строки по модулю 512 равнялась 0. То есть к числу **k** должны быть добавлены двоичное число длины **r** и **64** бита таким образом, чтобы эта сумма по модулю 512 равнялась 0:

(k + r +64) mod512=0;

$512 = 2^9$, R — двоичное представление числа К.

Например:

k=1024 бита, R=10, 64 — двоичное число из 1 0000…0 (63 нуля)

Общая длина сообщения M + прибавление (M`) разбивается на блоки по 512 бит и получается 512M или M блоков по 512 бит. В свою очередь каждый из блоков разбивается еще на 16 блоков по 32 бита, следовательно, длина в MD4:

M + M` = 512m, где 512 = 16*326.

Длина, соответствующая каждой части хэш-функции, также составляет 32 бита и чтобы получить функцию длиной 128 бит, нужно подсоединить 4 такие части. Эти части формируются на основе общего алгоритма, повторяющегося 3 раунда.

M: (k + 64 + r) mod 512 = 0, где (k + 64 + r) = 512m.

512 бит делятся на подблоки по 32 бита — основа для создания в дальнейшем функции хэширования.

Алгоритм создания хэш-функции MD4:

При создании хэш-функции используют следующие операции:

1. операция логического суммирования (\/ — “OR”, дизъюнкция);
2. операция логического умножения (/\ — “AND”, конъюнкция);

3. операция отрицания (“NOT”);
4. операция логического сдвига на S разрядов;
5. операция суммирования по модулю 2 (⊕).

Весь алгоритм состоит из 3-ех раундов. В каждом из раундов выполняется 16 шагов. Каждый шаг считает нелинейную функцию над 3-мя переменными из 4-ех. Итак, имеются 4 переменные: a, b, c, d. На каждом шаге выполняется операция над определенными 3-мя этими переменными. Кроме того, в каждом раунде используется своя функция. В 1-ом раунде используется функция F:

F (x, y, z) = x·y + $\overline{x}$ ·z, где “+” → “\/”, а “·” → “/\”.

Во втором раунде используется 2-ая функция — G, также из трех переменных:

G (x, y, z) = ((x·y) + (x·z) + (y·z)) — дизъюнкция 3-ех конъюнкций.

В третьем раунде:

H (x, y, z) = x ⊕ y ⊕ z.

x — входное сообщение, которое на каждом из 16 шагов в каждом раунде принимает одно из значений входного сообщения.

На первом шаге в первом раунде используются первые 32 бита последовательности — M_0. На втором — следующие 32 бита — M_1. На третьем — M_2. И так далее до 16-го раунда — M_{15}. таким образом:

X = M_0, M_1, M_2, M_3... M_{15} = 512.

Значение Y не меняется в каждом из раундов. В первом раунде Y = 0 (32 раза повторяющийся ноль), во втором раунде значение Y представляется в 16-ой форме: 5A827999, где 5 → 0101 (4 разряда), A → 1010 и так далее. В третьем раунде значение Y = 6ED9EBEF.

Переменная z определяет очередность каждого из 16-ти подблоков в каждом раунде. В 1-ом раунде — очередность от 0 до 15: z = 0, 1, 2…15 — соответствует нумерации блоков в каждом из 512 символов.
Во 2-ом раунде: z = 0, 4, 8, 12, 1, 5, 9, 13,…
В 3-м раунде: z = 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15.

Для реализации алгоритма инициируются также 4 переменные:
a = 01234567_{16} — 16-ая форма записи 2-го числа;
b = $89ABCDEF_{16}$
c = $FEDCBA98_{16}$

$d = 76543210_{16}$

Переменные a, b, c, d в ходе выполнения алгоритма изменяют свои значения. Хеш-функцией будут значения этих переменных, соединенные по методу конкатенации (“связывания”) в последовательности a, b, c, d.

Основной цикл состоит из 3-ех похожих раундов, в каждом раунде выполняется 16-ть различных операций, каждая операция подсчитывает функции F (на первом раунде), G (на втором раунде) и H над тремя переменными из массива a, b, c, d.

На каждом шаге выбираются 3 определенные переменные. Общий вид операции, например, в 1-ом раунде записывается так:

FF (a, b, c, d, M_j, S), $j \in [0,15]$, где M_j — блок данных, S — сдвиг.

$a = b + (a + F (b, c, d) + M_j <<< S)$, запись “<<< S” означает сдвиг на S символов вправо.

Алгоритм прибавляет результат вычисления к 4-ой переменной a. Полученный результат циклически сдвигается на S разрядов вправо и добавляется к одной из 4-ех переменных (в нашем случае b). Результат помещается в одну из переменных (в нашем случае a), которая будет использоваться на следующем шаге.

Количество шагов, на которое сдвигается результат: 7 на 1-м шаге, 12 на 2-м шаге, 17 на 3-м.

На втором шаге функция имеет вид:

GG (d, b, c, a, M_j, S)
$d = a + (d + F (a, b, c) + M_j <<< S)$

Хэш-функция MD5:

Основные отличия:

1. добавляем 4-ый раунд — в нем используется нелинейная функция:

$I (x, y, z) = y + (x \vee \overline{z})$;

2. на каждом раунде и шаге используется уникальная константа, которая обозначается, как T. Эта константа добавляется к результату:

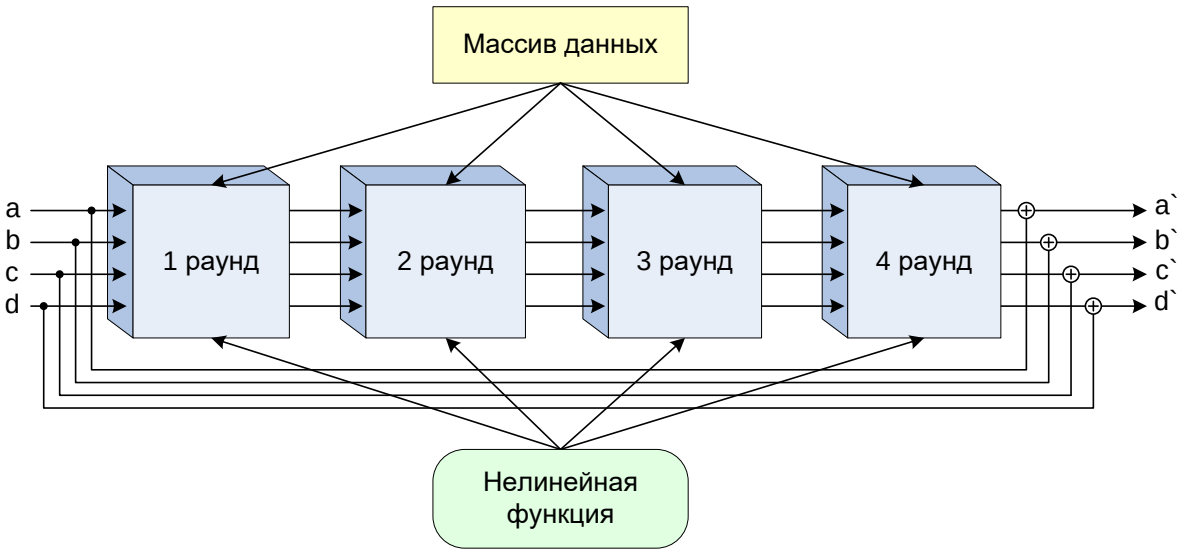
FF (a, b, c, d, M_j, t_i, S)
$a = b + (a + F (b, c, d) + M_j + t_i <<< S_j)$

3. функция G на втором раунде заменена на менее симметричную функцию:

a (x, y, z) = ((x·z) + (y· $\bar{z}$))

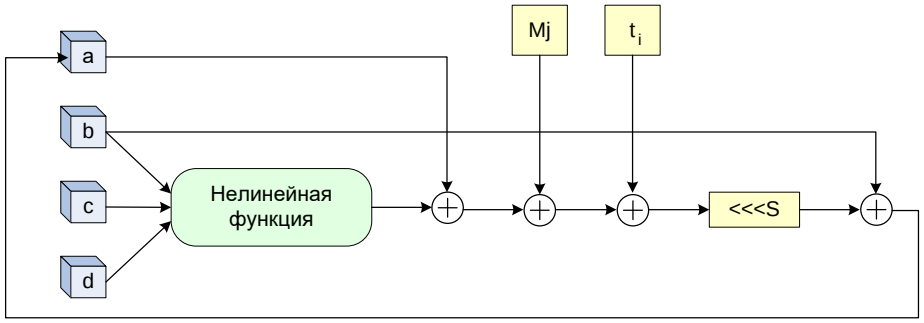
4. результат каждого раунда добавляется к предыдущему;
5. порядок следования от блоков изменяется во 2-м и в 3-м раундах.

Схематично:



H (M) = $h_1 \| h_2 \| h_3 \| h_4$;

Одна из операций на шаге выглядит следующим образом:



Переменная t на 1-м шаге 1 раунда: t_{11} = d76aa178;
На 2-м шаге первого раунда: t_{12} = e8C7B756;

На i-том шаге константа $t_i = 2^{32} \cdot \sin(i)$, где i [рад].

Хэш-функции класса SHA.
“Secure Hash Algorithm” — безопасный алгоритм хеширования.

Национальный институт стандартизации и технологий (NIST) и агентство национальной безопасности разработали алгоритм хеширования для использования в цифровой подписи DSS на основе DSA (Digital Signature Algorithm).

Особенности:

1. длина хеш-функции 160 бит в отличие от предыдущих 128-ми;
2. все предыдущие операции такие же как и в MD. То есть все сообщение делится на блоки длиной 512 бит;
3. используются 4 раунда. Введена своя 5-ая переменная. Имеем, соответственно, a, b, c, d, e. Длина каждой из них 32 разряда. Итого, $32 \cdot 5 = 160$ бит. В каждом раунде реализуется 20 шагов.

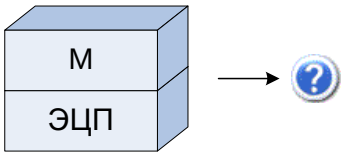
Из всех хеш-функций, SHA1, является наиболее криптостойкой.

Сравнительная таблица:

Алгоритм вычисления	Длина (бит)	Скорость хеширования (Кбит/с)
MD4	128	236
MD5	128	174
SHA1	160	75
ГОСТ	256	11

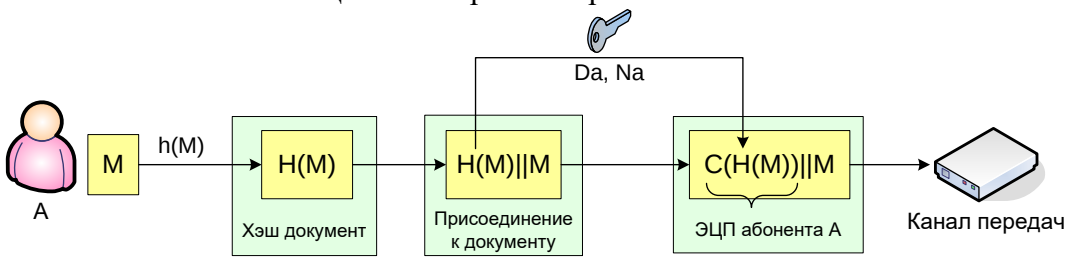
ЭЦП на основе хэш-функции.

Наиболее простой способ генерации и использования ЭЦП состоит в сообщении, которое нужно подписать на основе хэш-функции.

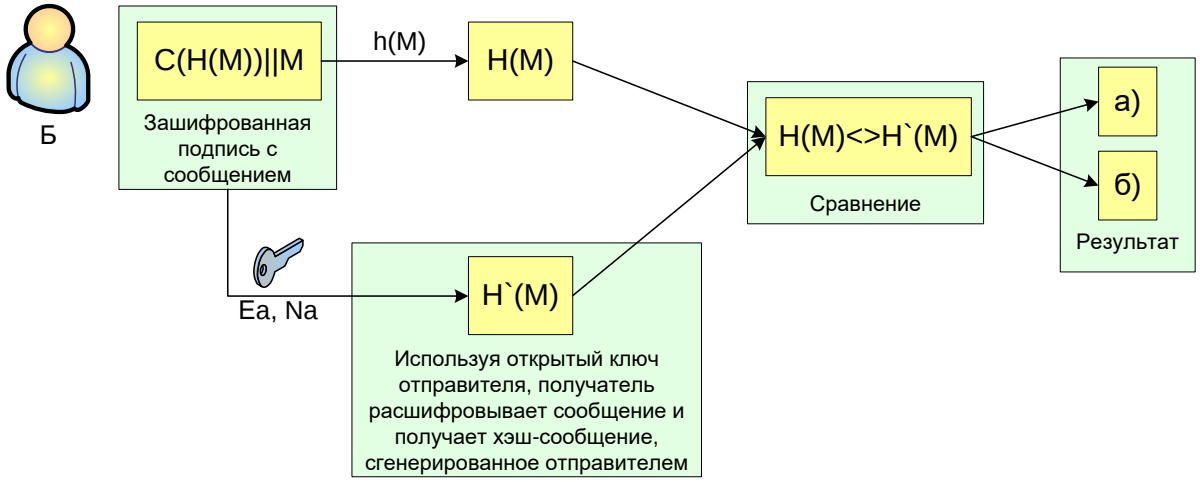


1. Использование ЭЦП для верификации документа.

Используемая подпись подтверждает, что полученный документ не видоизменялся. Сообщение пересылается в незашифрованном виде. Обработке подвергается только часть с ЭЦП. На стороне отправителя:



Возникает вопрос: “Что делать дальше с этим хеш-значением?” Оно может быть зашифровано с помощью тайного ключа отправителя (N_A). Таким образом, в результате получается зашифрованный хэш C(H(M)||M). На стороне получателя:



а) *они равны друг другу*. В этом случае Б делает вывод:

- сообщение действительно отправлено А, то есть А — действительно является автором, то есть происходит идентификация авторства;
- полученное сообщение М соответствует тому, которое было составлено отправителем и не претерпело никаких изменений.

b) *не равны*. Тогда Б делает заключение:

- либо А не является автором;
- либо полученный документ в результате пересылки был модифицирован 3-й стороной;
- либо и то и другое.

Возникает вопрос: “Почему документ был модифицирован?”. Вследствие того, что сообщение передавалось с открытым ключом и по открытым каналам.

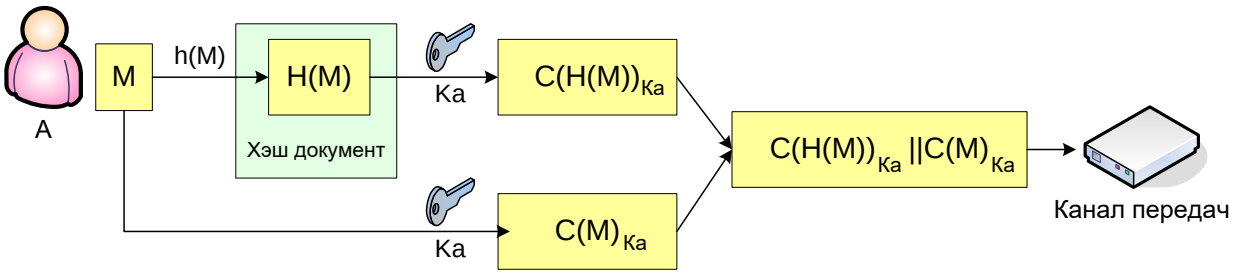
2. Метод использования ЦП с шифрованием сообщения.

Шифрование сообщения может быть основано как на симметричном, так и на асимметричном методе.

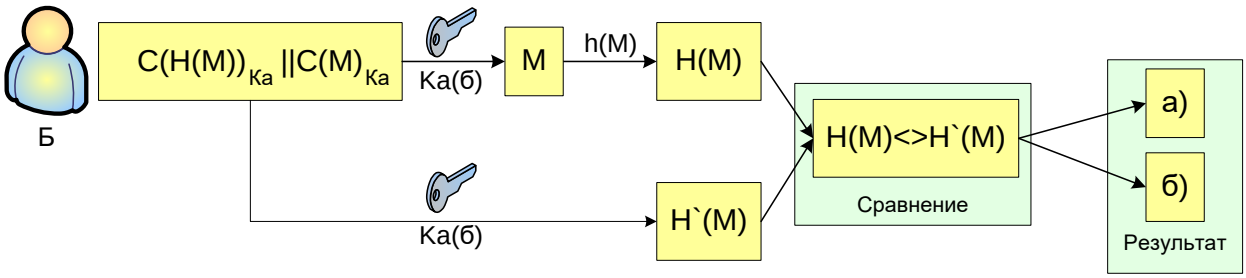
Шифрование сообщения с использованием симметричных систем.

Как и раньше создается хэш-значение сообщения М. На основе тайного ключа абонента А, создается $C(H(M))_{Ka}$ или создается ЭЦП. Сообщение М на основе того же ключа шифруется и в канал передаются соединенные зашифрованные тем же ключом сообщение и ЦП.

На стороне отправителя:



На стороне получателя:



Используя ключ $К_{А,}$ Б получает сообщение М, хэш-функцию и далее сравнивает эти два значения, как и в предыдущем случае.

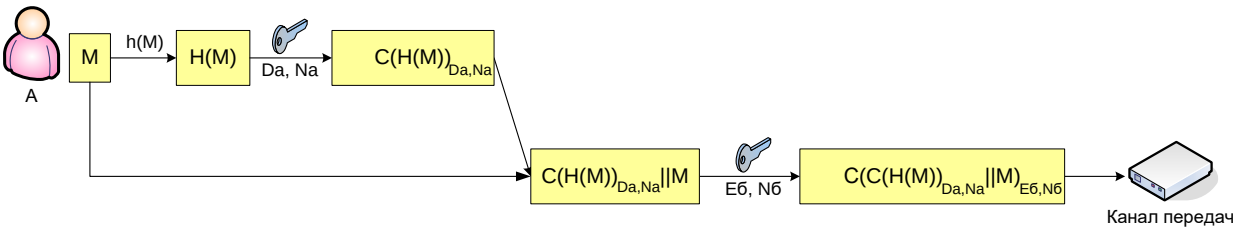
Этот метод с точки зрения криптостойкости практически не отличается от метода обычного симметричного шифрования, ЦП играет в основном формальную роль, если ключ известен только А и Б.

ЭЦП на основе асимметричной системы (данная методика наиболее распространена).

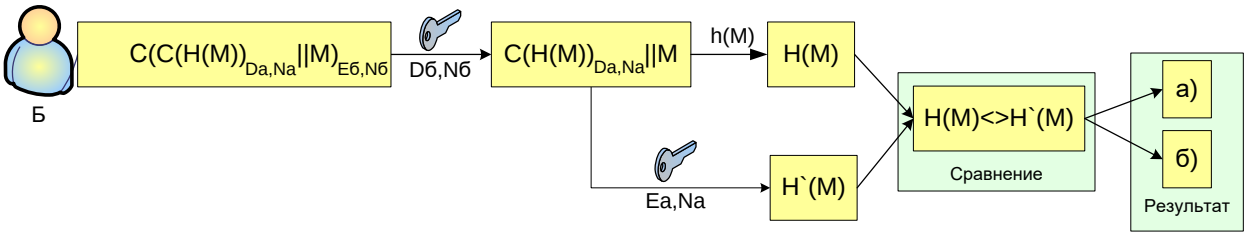
Используя известную хэш-функцию, создается дайджест (хэш-функция) сообщения. Далее, используя тайный ключ отправителя, создается ЭЦП, шифруется и присоединяется к документу (М). Все это сообщение далее шифруется стандартным образом, используя открытый ключ получателя. В конце создается шифрограмма: $C(H(M)_{Da,Na} \| M)_{Eб,Nб.}$ (*)

Как видно из формальной записи итоговый электронный документ формируется на основе 2-ух пар ключей: тайного ключа отправителя, для генерации ЭЦП, и открытого ключа получателя, для шифрования составного (сообщение + подпись) документа.

На стороне отправителя:



На стороне получателя:



Получатель Б получает сообщение (*), используя свой тайный ключ ($E_б$, $N_б$), Б расшифровывает составной документ и получает $C(H(M))_{Da,Na}||M$. Дальнейшие операции основываются на рассмотрении п. №1. А именно: используя известную хэш-функцию, создается хэш-значение полученного и расшифрованного сообщения. Полученные хэш-значения сравниваются между собой:

a) *они равны друг другу*. Устанавливается подлинность и авторство документа.

b) *не равны*. Следовательно, возможны 3 ситуации, описанные выше.

На практике наиболее часто используется именно этот метод. Основой этого метода является несимметричный метод шифрования (метод Диффи-Хеллмана). Алгоритм генерации на основе асимметричных систем разрабатывал El-Hamal. В Беларуси используется стандарт Shnorr, метод RSA. В асимметричных системах, если процесс расшифрования закончился успехом, то устанавливается авторство.

Выводы:

1. Криптографическое преобразование (шифрование данных) является наиболее эффектным средством повышения уровня конфиденциальности, переданных или хранимых данных.
2. Известны 2 общих методов шифрования:

- С тайным ключом (симметричный метод);
- С публичным ключом (асимметричный метод);

3. На основе этих методов шифрования может генерироваться ЭЦП, подсоединяться к основному документу как элемент, удостоверяющий авторство и подлинность документа.
4. ЭЦП создается 2-мя общими методами:

- Простым шифрованием (позволяет установить авторство на основе ключей);

- С использованием хэш-функций;

5. ЭЦП, подсоединенная в зашифрованном виде к основному документу (М), и зашифрованное хэш-значение, образуют составной документ, который в свою очередь также может быть зашифрован.
6. Характеристикой эффективности методов шифрования является уровень криптостойкости (определяет вероятность взлома шифра за определенное время) и скоростью шифрования или расшифрования.
7. В сравнении с 2-мя ранее рассмотренными методами преобразования (помехоустойчивость и сжатие) методы криптопреобразования характеризуется тем, что:

$\{X_k\} \neq \{Y_n\}$;
k = n — длина последовательности до преобразования не меняется и после преобразования, за исключением ЭЦП: k < n.

Стеганографические методы повышения уровня конфиденциальности информации.

Стеганография — осаждение (размещение) одной информации в другой, называемой “контейнером”.
Методы стеганографии предусматривают следующие контейнеры:

- Текст;
- Звук;
- Изображение.

Методы осаждения информации в текстовых документах:

- Через использование разного числа пустых знаков между символами;
- Знакоосаждение на нужных позициях. Знаки по контексту не меняют логики текста;
- Изменение высоты отдельных букв.

Методы осаждения информации в звуках, на основе звуковой информации. В высококачественных системах, где используется больше 10 символов квантования по уровню, осаждение происходит в младших значимых битах.

Тот же принцип используется в размещении информации в изображениях. Осаждение нужной информации в младших значимых битах.

Методы стеганографии широко используются для защиты программного обеспечения от несанкционированной модификации и главным образом для доказательства авторства.

С точки зрения современной информационной технологии — контейнером для может быть текстовые, звуковые и графические.

Характеристика деструктивных программных средств, методов их внедрения в информационные системы и методов борьбы с ними.

1. Деструктивные программные средства инфицируют используемые программные обеспечения через операционные системы, через системы управления БД, через сетевое программное обеспечение. Инфицирование основывается на поиске и использование брешей в такой системе. Термином “брешь” характеризуются недостатки операционной системы вследствие недосмотра создателей. С появлением каждой новой операционной системы, количество таких брешей становится все больше и больше. Для поиска таких брешей были созданы специальные средства — сканеры портов “sniffer”. Первоначально sniffer’ы создавались для обнаружения деструктивных программных средств. Сейчас используются в корыстных деструктивных целях.
2. Попадание инфицированных деструктивных средств *через систему управления*. В общем случае есть ОС, система управления БД и прикладные программы. Как правило, доступ пользователя к целой системе осуществляется через ОС с помощью пароля и логина. Чтобы получить логин, нужно пройти идентификацию повторно и в системе управления. Некоторые системы управления разрешают вхождение в БД после получения подтверждения. Если пользователь получил разрешение, он автоматически получает доступ к БД.
3. *Через сетевое программное обеспечение*. Сетевое программное обеспечение является часто используемым фактором вхождения интруза в систему.

Основные методы защиты от попадания интруза через сеть:

a. Сегментация сети;

b. Использование брандмауэров (firewall, контролирующих входной трафик) и спамовые программные обеспечения.

Общая характеристика информационных систем с точки зрения их безопасности.

Программное обеспечение, компьютерная сеть и телефонная сеть относятся к информационным системам. В начале 90-ых годов военный департамент “Department of Defence” начал создавать стандарты и требования к безопасности систем. Эти требования и стандарты стали известны как “Red Book” (“Красная книга”) — свод правил и требований к информационным системам с точки зрения безопасности. Позднее, эти требования стали общедоступными и их стали рассматривать как 1-й стандарт в области информационной безопасности систем. Эти требования стали известны как “Orange Book”(“Оранжевая книга”). В соответствии с оранжевой книгой все системы с точки зрения безопасности делятся на классы: D, B, C, A. Наименее безопасной является система, относящаяся к классу D. Наиболее сильную степень защиты имеет класс A.

Чуть позднее, на основе оранжевой книге была опубликована книга “Lavender Book” (“Лиловая книга”) — содержит свод правил БД с точки зрения безопасности.

Красная и лиловая книги — общеизвестный свод правил и норм безопасности. На их основе разработаны национальные стандарты, в том числе и в Беларуси.

В соответствии с этими правилами безопасность любой системы анализируется с точки зрения реализованного в ней метода разграничения доступа объектов к информационным ресурсам. Объектом может быть пользователь, устройство или другой процесс.

В соответствии с этими книгами существуют 2 класса методов разграничения доступа:

1-й метод: обязательный (mandatory);
2-й метод: по усмотрению (discretional).

1-й метод используется в основном в системах, характеризующихся статичностью структуры и статичностью обрабатываемой информации. Методы этого класса основываются на том, что каждый объект данных характеризуется установленным уровнем тайности или конфиденциальности.

Существуют следующие уровни конфиденциальности:

СС (совершенно секретно), ДПС (для служебного пользования), С (секретно) и НС (несекретно).

В свою очередь каждый объект (пользователь, устройство или процесс) также должен характеризоваться с точки зрения безопасности системы правом доступа к этим системам.

Пусть, пользователь А имеет доступ, характеризующийся как, С (секретно). В соответствии с требованием класса “ обязательный ”, такой пользователь имеет право доступа к информационным объектам, соответствующим уровню безопасности не выше, чем уровень С.

2-й класс методов. В основном, эти методы нашли применение для разграничения доступа к БД. И этот доступ основывается на строгих математических моделях. Наиболее известная из таких математических моделей является так называемая модель RBAC.

RBAC (Role Based Access Control) — управление доступом, основанное на ролях.

Что такое роль? Она регламентирует доступ пользователям к отдельным элементам БД.

RBAC — наиболее простая модель разграничения доступа пользователей к элементам БД.

Контроль доступа осуществляется в основном для передачи, хранения, частично обработки информации.

Контроль доступа “по усмотрению” предусматривается в основном для систем, реализующих БД.

Среди 4-ех классов, характеризующих безопасность систем, класс D объединяет системы. Системы класса С делятся на 2 подкласса: С1 и С2.

В этих системах, как правило, реализуется политика безопасности на основе методов 2-го класса. Причем системы подкласса С1 являются менее защищенными. Эти системы требуют разделения данных и пользователей. Практически это означает, что признается возможность существования совместных данных, а также возможность существования данных, находящихся в собственности конкретного пользователя. Система подкласса С2 требует реализации дополнительных механизмов контроля, что главным образом основывается на мониторинге (аудите) событий и изолировании процессов. Примером системы, относящейся к подклассу безопасности С2, является ОС Windows NT 4.0.

Системы класса B реализуют политику безопасности на основе обязательных методов. Системы этого класса в свою очередь делятся на 3 подкласса:

- B1;
- B2;
- B3.

Из которых системы подкласса B1 наименее защищенные. Системы подкласса B1 требуют, чтобы каждый объект данных был обеспечен соответствующим уровнем безопасности, начиная от СС > С > ДПС > НС.

Системы подкласса B2 дополнительно требуют разработки формальных процедур и правил, описывающих порядок доступа объектов к информационным ресурсам, то есть требуют разработки какой-то математики. Кроме того, требуют, чтобы были ликвидированы закрытые каналы передачи, чтобы они были трансформированы в общедоступные каналы с реализацией соответствующих методов повышения уровня конфиденциальности передаваемых через них информации.

Системы подкласса B3 требуют реализации механизмов аудита, а также использования лиц физических, ответственных за реализацию политики безопасности.

Большинство коммерческих систем управления БД относятся к подклассу B1.

Системы класса A являются наиболее защищенными, они дополнительно требуют разработки развитого математического аппарата для описания процессов этих систем с точки зрения безопасности. Кроме того, должны существовать механизмы контроля реализации уровня безопасности.

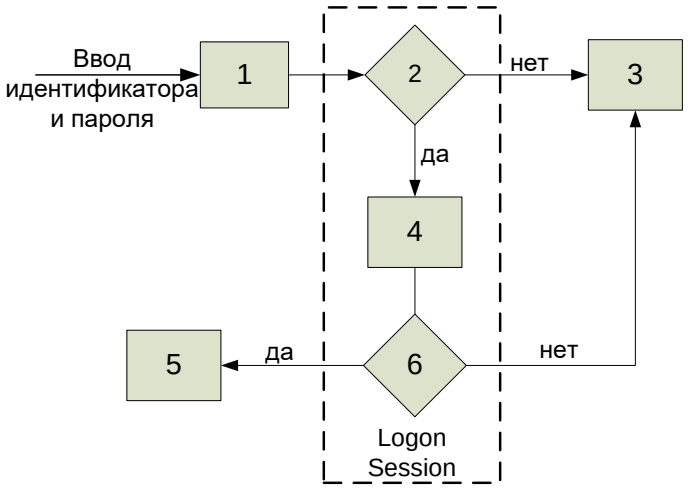
Защита программного обеспечения.

Наиболее эффективный метод защиты основан на использовании пароля. С формальной точки зрения пароль ― особым образом сгенерированная последовательность символов на основе одного или нескольких алфавитов и предназначенная для реализации процедуры идентификации объектов, чаще всего пользователя, реализующего механизм доступа к информационной системе (к ПО).

В связи с анализом пароля нас интересует процедура анализа самого пароля, а также свойства пароля: использованный алфавит, длина пароля и критическое время его использования. Критическое время характеризуется “безопасным временем использования пароля”.

Механизм анализа пароля.

Этот механизм можно представить следующей обобщенной схемой:



1 – ввод;

2 – проверка идентификатора и пароля на предмет содержания ID и пароля в базе (в Windows в базе SAM (Security Account Message));

3 – если ID и пароль не найдены, то выводится сообщение об отказе в доступе;

4 – устанавливаются полномочия пользователя с введенным паролем;

6 – имеет ли он полномочия на работу с ресурсами?

Если нет, то следует отказ в доступе (3). Если да, то

5 – разрешение в использовании ресурса.

Процедура идентификации и аутентификации пользователя.

Анализируя такой механизм, принимаем во внимание особенности:

1. На каком этапе взаимодействия пользователя с системой инициируется Logon Session. В различных системах используются разные способы реализации процессов: на основе BIOS, MBR.
2. Какие параметры используются для идентификации пользователя:

- Набор пароля на клавиатуре;
- Однако в информационных системах используются и другие параметры, например, биометрические (оттиски пальцев, сетчатка глаза) и антропометрические (рисунок на ладонях, овал лица),;

- Либо могут использоваться физические принципы, основанные на использовании ключевой информации, размещенной на внешних носителях: SMART-карты, магнитные карты и др.

Основным методом для идентификации является использование пароля. Анализируются эффективность и безопасное время использования пароля.

Обозначим длину пароля как S.

H → {Abc*!123}, то есть S = 8.

Алфавитом являются символы верхнего и нижнего регистра латинского алфавита, цифры и специальные знаки, то есть в этом примере используются все 4 набора исходных символов.

A $\{a_i\}$ – алфавит, состоящий из фиксированного набора символов.

$i \in [1, N]$, N – мощность, количество символов, входящих в этот алфавит.

Если N – это мощность данного алфавита, то общее число паролей, которые могут быть сгенерированны на основе этого алфавита, обозначатся как $I_H = N^S$.

Пример:

Имеем двоичный алфавит, состоящий из 0 и 1.

A {0, 1}, N = 2, S = 2, то $I_H = N^S = 2^2 = 4$ (00, 01, 10, 11).

Если алфавит составлен на основе символов:

A {a, b, … z}, N = 26, S = 8, то $I_H = N^S = 26^8 = 208827064576$.

Безопасное время использования сгенерированного пароля фиксированной длины, обозначается как t_h. Принято считать, что оно должно быть не больше, чем $½ (I_H \cdot T)$.

$t_h = ½ (I_H \cdot T)$, где значение параметра T рассчитывается как:

T = E/R, где E – количество символов в комбинации, которая сгенерированная хакером, для лобовой атаки системы (с целью проверки всех возможных комбинаций).

$E = S + S_{сл}$, принято считать, что $S_{сл}$ составляет 40-60% от общей длины последовательности, если длина пароля S состоит из 5-15 символов.

R – скорость передачи информации по используемым каналам. R может выражаться с физической точки зрения как [сим/сек] или [бит/сек]. Таким образом, видно, что T – есть время анализа одного пароля, который генерирует хакер, высылает через сеть и пробует с его помощью взломать защиту.

Чтобы перехватить, например, Троян, вся Logon Session должна быть изолирована.

Пример:

Используется алфавит, мощность которого N = 5 символов, S = 6 символов, скорость передачи R = 3 [Кбит/сек].

T = E/R ([сим]/[бит /сек]). Для того чтобы вычислить T, необходимо перевести символы в бит или наоборот.

Предполагаем, что, как и при использовании кодов ASCII 1 символ = 8 бит.

$S_{сл}$ = 4 символов. E = 6 + 4 = 10 символов (либо 80 бит).

R = 3 [Кбит/сек] = 375 сим/сек.

$t_h = ½ (I_H \cdot T) = t_h = ½ (5^6 \cdot 10/375_{сек}) = ½ (15625 \cdot 0.027_{сек}) = 211$ (сек)

Андерсен проанализировал ситуацию с паролем с другой стороны. Он попробовал ответить на вопрос: “Каким выбрать пароль, чтобы он не был сломан при фиксированных параметрах и характеристиках систем за определенное время?”.

Стоит задача: как для заранее установленного времени выбрать пароль.

Предполагаем, что P – это вероятность того, что пароль будет сломан за месяц. Нижняя граница для вероятности обозначается как P_0.

$P_0 \leq P$ - вероятность того, что этот пароль не будет сломан за M месяцев.

$P_0 = L/m$, где L – количество попыток взлома пароля за M месяцев. $m \rightarrow I_H$.

В свою очередь, параметр $L = L_1/ L_2$, где L_1 — есть количество символов, которые можно передать через используемые каналы за M месяцев, а L_2 — количество символов в одной пересылаемой комбинации, $L_2 \rightarrow E$.

В свою очередь, $L_1 = R \cdot M \cdot 30 \cdot 24 \cdot 3600$, где R — скорость передачи, M — количество месяцев.

Таким образом, Андерсен записал значение $P_0 = \frac{2.6 \cdot 10^6 \cdot R \cdot M}{E \cdot N^S}$.

$P \geq \frac{2.6 \cdot 10^6 \cdot R \cdot M}{E \cdot N^S}$.

Формула Андерсена: $N^S \geq \frac{2.6 \cdot 10^6 \cdot R \cdot M}{E \cdot P}$

Отсюда, по известным значениям нужно выбрать размер алфавита и длину пароля, чтобы с заданной вероятностью этот пароль не был сломан за несколько месяцев.

Пример:

$P = 10^{-3}$;

M = 3;
R = 10 (сим/сек);
E = 20 (сим);
N = 26 (сим);
S = 6 (сим);

$26^6 \approx 3.089 \cdot 10^8 \leq 3.9 \cdot 10^9$. Это означает, что при выбранном размере алфавита и длине пароля, необходимое условие неравенства не выполняется. При S = 7 (сим):

$26^7 \approx 8.03 \cdot 10^9 \geq 3.9 \cdot 10^9$. Выполнение условия $N^S \geq \frac{2.6 \cdot 10^6 \cdot R \cdot M}{E \cdot P}$ означает, что для выбранного алфавита, с вероятностью P = 10^{-3} пароль с длиной 7 символов не будет сломан за 3 месяца.

Протокол "Керберос" (Protocol Kerberos).

Протокол "Керберос" предназначен для пересылки зашифрованного сообщения по открытым каналам на платформе ОС Windows и опирается на технологию транзитивных "доверительных отношений".

Доверительной стороной всегда выступает служба "Керберос", которая управляет выдачей ключей любым 2-м сторонам А и В, участвующим в обмене информацией. Протокол "Керберос" в основе своей опирается на протокол Нидхэма-Шрёдера (R. Needham-M. Schröder) и базируется на симметричном шифровании данных.

Сущность протокола Нидхэма-Шрёдера:

На первом этапе передается сообщение от А к В, включающее ключ Ra — случайное число, генерируемое А. С — посредник.

С, генерирует случайный сеансовый ключ К, затем он шифрует секретным ключом, общим для С и В, сообщение, включающий этот ключ и имя А: ЕВ (К, А). Затем С шифрует секретным ключом общее для С и А случайное число RA, KA, B: EA (RA, KA, B, EB (K, A)).

А расшифровывает сообщение и извлекает из него К, кроме того, абонент А убеждается, что случайное число сгенерированное им, совпадает со значением, полученным от С. Это является подтверждением того факта, что документ отправлен именно посредником, то есть подтверждается его аутентификация. Затем А отправляет В сообщение, которое зашифровал С. А: ЕВ (К, А) → В.

В, зная ключ, расшифровывает полученное сообщение и извлекает из него случайное число, ключ К, который является сеансовым ключом. Затем В генерирует случайное число RB. Это число он шифрует сеансовым ключом К. В: ЕК (RB) → А.

А расшифровывает полученное сообщение, известным ему ключом К. Затем А, генерирует число RB-1, шифрует это число с помощью того же ключа KA и посылает это зашифрованное сообщение В. А: ЕК (RB-1) → В.

B расшифровывает полученное сообщение с помощью K и удостоверяется, что ему было прислано сообщение RB-1, тем самым абоненты A и B, посредством C завершают взаимную аутентификацию. Все эти генерации RA, RB, RB-1 преследуют следующую цель: уменьшается вероятность того, что четвертая сторона D расшифрует ключ KA, анализируя вложенное сообщение.

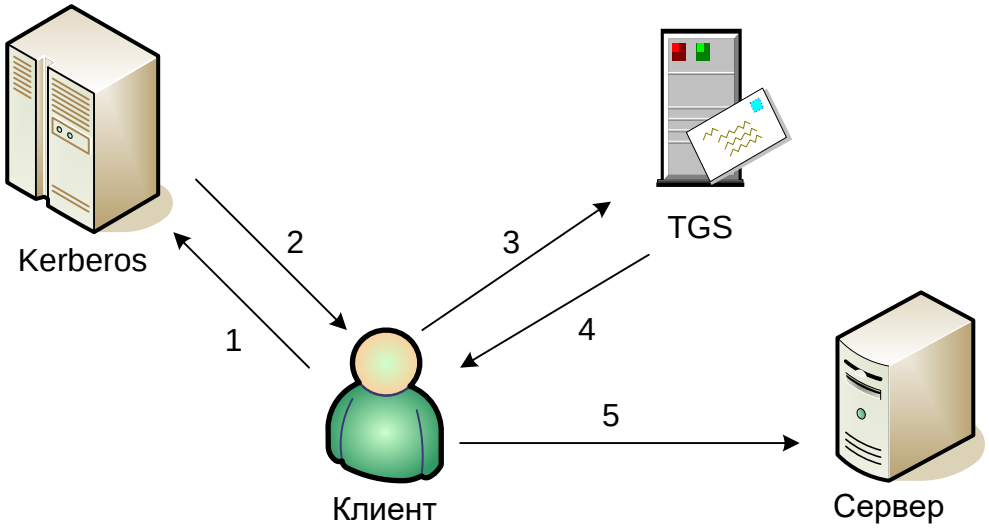
Протокол Керберос в базовом варианте предполагает, что A и B имеют общие ключи с C. Дополнительно к протоколу Нидхэма-Шрёдера в 5-ой версии в качестве одной из основных единиц информации, используется метка времени t — продолжительность сеанса или продолжительность времени, в течение которого действует ключ.

Установленная в сети TCP/IP служба Керберос, является доверенной стороной (C). Пользователю предоставляется возможность работать на различных компьютерах, подключенных в сеть. Реализация протокола Керберос подразумевает соединение Клиента с Сервером. Таким образом, Клиент может быть пользователем или процессом, например, доступ к принтеру. Протокол Керберос, как и большинство протоколов, обеспечивающих безопасность, разработан в Массачусетском технологическом университете в проекте Athena.

Основой Керберос является БД Клиентов и их секретных ключей. Для пользователей такой ключ является зашифрованным паролем. Сетевые службы, которые требуют аутентификацию, должны зарегистрировать в Керберос также и свои секретные ключи.

Так как Керберос знает все секретные ключи, он может убеждать одни объекты в подлинности других. Керберос создает сеансовые ключи, которые выдаются Клиенту и Серверу, и никому больше. Именно сеансовый ключ используется для шифровки сообщения, которыми обмениваются две стороны, по окончанию сеанса этот ключ уничтожается. Для шифровки используется алгоритм DES.

Для организации канала связи Клиент запрашивает у Керберос разрешение на обращение к службе организации таких сообщений, эта служба называется Ticket Grating Service (TGS) — служба выделения мандата.



1 — клиент запрашивает Керберос разрешение на обращение к службе TGS.

2 — после анализа предоставленных документов о возможности организации сообщения между Клиентом и Сервером, Керберос выдает Клиенту соответствующее разрешение.

3 — пользуясь разрешением службы Керберос, Клиент запрашивает TGS о выделении ему мандата на организацию канала между Клиентом и Сервером.

4 — получение такого мандата.

5 — Клиент пересылает соответствующее сообщение серверу.

C— Клиент (Client).

S – Сервер (Server).

A — сетевой адрес Клиента (Address) — имя Клиента.

v — временная метка, содержащая начальное и конечное время действия мандата.

t — просто метка времени, соответствующая периоду времени, в течение которого действует сеансовый ключ.

$\mathbf{K_x}$ — секретный ключ объекта X.

$\mathbf{K_{x,y}}$ — сеансовый ключ для организации сеанса между X и Y.

$\mathbf{\{m\}_{Kx}}$ — сообщение m, зашифрованное ключом K_x.

$\mathbf{T_{x,y}}$ — мандат, выданный X на использование Y.

$\mathbf{A_{x,y}}$ — аутентификатор, выданный X для Y, то есть информация, с помощью которого Y аутентифицирует X.

Керберос использует 2 типа удостоверений:

- Мандаты (для безопасной передачи Серверу данных о личности Клиента).
- Аутентификаторы (это дополнительная информация, предъявляемая вместе с мандатом).

Мандат Керберос имеет форму:
$\mathbf{T_{c,s} = S, \{C, A, v, K_{c,s}\}K_s}$

Мандат, таким образом, содержит:

- Имя Клиента (**C**).
- Его сетевой адрес (**A**).
- Имя сервера (**S**).
- Метку времени (**v**).
- Сеансовый ключ ($\mathbf{K_{c,s}}$).

Эта информация шифруется секретным ключом Сервера $\mathbf{K_s}$.

Клиент не может расшифровать мандат, поскольку он не знает секретный ключ $\mathbf{K_s}$, но он может предъявить его Серверу, как доказательство его аутентичности, то есть прочитать либо изменить мандат, ни Клиент, ни кто-либо иной не может.

Аутентификатор имеет следующую форму:
$\mathbf{A_{c,s} = \{C, t, Ключ\}\ K_{c,s}}$

Клиент создает аутентификатор на каждый сеанс, Ключ — является просто ключом и необязательным дополнительным элементом сеанса и все эти данные шифруются общим ключом, известным Клиенту и Серверу $\mathbf{K_{c,s}}$. В отличие от мандата, аутентификатор используется только один раз.

Все стрелки 1-5 могут быть записаны в формализованном виде:
1 — **C, TGS**
2 — $\mathbf{\{K_{c,TGS}\}K_c}$; $\mathbf{\{T_{c,TGS}\}K_{TGS}}$
3 — $\mathbf{\{A_{c,s}\}K_{c,TGS}}$; $\mathbf{\{T_{c,TGS}\}K_{TGS}}$
4 — $\mathbf{\{K_{c,s}\}K_{c,TGS}}$; $\mathbf{\{T_{c,s}\}K_s}$
5 — $\mathbf{\{A_{c,s}\}T_{c,s}}$; $\mathbf{\{T_{c,s}\}K_s}$

Защита регистра.

Регистр — БД о конфигурации компьютеров в сети. Регистр реализует примерно те же функции, что и файлы .ini в системе MS DOS. В регистре есть множество поддеревьев, состоящих из ключей, которые собственно содержат информацию о компьютере, о пользователях, о группе пользователей. Регистр состоит из 5 ключей, каждый из которых начинается со слова H_Key. Один из важнейших ключей — H_Key_Local_Machine, содержащий информацию о локальной системе, в том числе о бюджете пользователей и об элементах локальной политики безопасности, которая в свою очередь состоит из подключей, важнейшим из которых является SAM. Эта БД с функциями менеджера бюджета

безопасности, которая содержит информацию об идентификаторах и паролях пользователей и групп пользователей. Причем эта информация хранится в зашифрованном виде и преобразовывается на основе хеширования данных с изменением механизма MD.

Подключ Security содержит информацию о правах или привилегиях пользователей, кто из пользователей имеет какое право для доступа к каким объектам.

Подключи содержаться в подкаталогах ОС: …\System 32\Config. Доступ к этим ключам запрещен для всех пользователей.

Общая модель безопасности ОС опираются на следующие элементы:

1. Процесс анализа идентификатора и пароля (Logon session).
2. Local Security Authority (LSA) распорядитель локальной безопасности, который должен гарантировать то, что каждый пользователь, зарегистрированный системой, имеет право доступа к ней. Именно алгоритм LSA вырабатывает так называемый маркер безопасного доступа: Security Access Token. Security Access Token вырабатывается в системе, в каждом сеансе работы пользователя с системой. SAT управляет локальной политикой безопасности, процедурой аутентификации пользователя, политикой аудита событий, регистрируя эти события в соответствующем журнале.

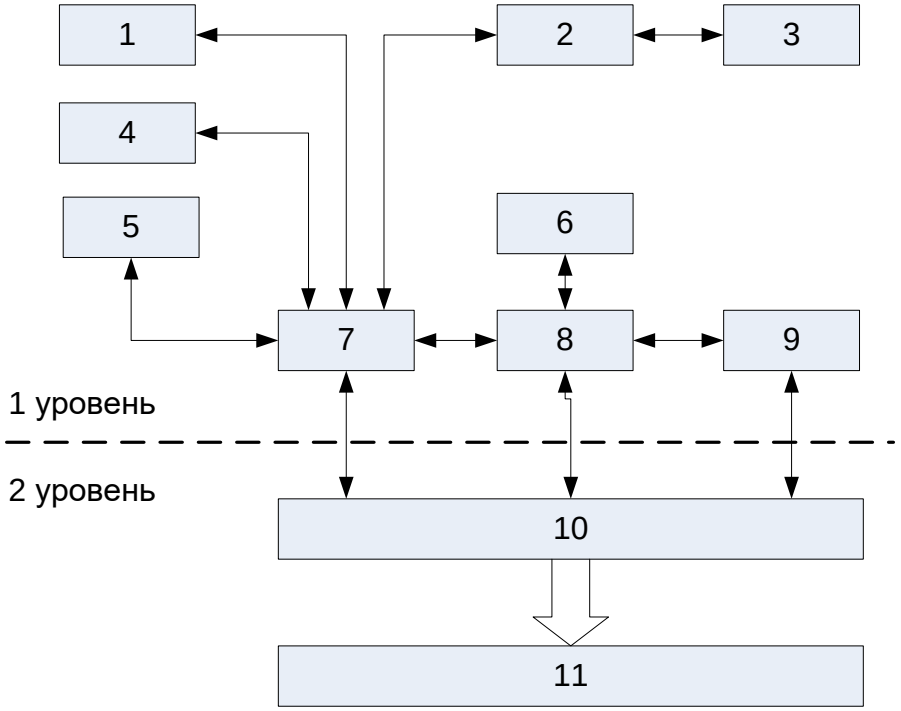
Маркер безопасного доступа содержит идентификатор доступа или идентификатор группы, к которому принадлежит данный пользователь, а также содержит информацию о правах доступа к объектам ОС.

Диспетчер бюджетов безопасности.

Security Accounts Monitor (SAM_1) — этот компонент предназначен для взаимодействия с БД бюджетов пользователей.

Security Reference Monitor (SRM) — диспетчер безопасности. Это компонент, который отвечает за контроль, установление и реализацию прав доступа к объектам.

Схематично политику безопасности можно представить следующим образом:



1 — процесс входа в систему Logon session.
2 — Security Accounts Monitor.
3 — БД SAM
4 — БД политики безопасности.
5 — журнал или БД аудита событий.
6 — приложение Win 32.
7 — LSA (диспетчер локальной безопасности).
8 — подсистема Win 32.
9 — другие подсистемы.
10 — сервисы, исполнительные части систем, например, драйверы и другие.
11 — Hard Ware — аппаратные средства.

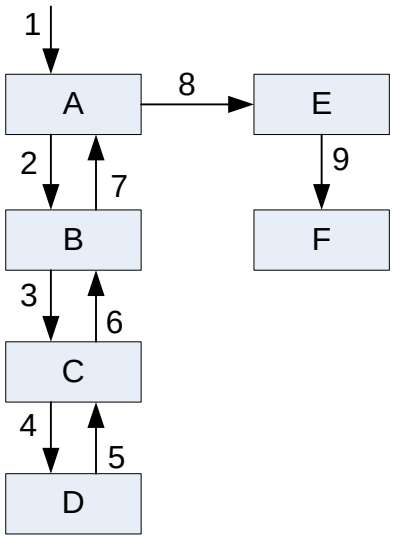
Разделим на два уровня:

1-й уровень — уровень пользователя (User mode).
2-й уровень — уровень ядра (Kernel mode).

Маркер безопасного доступа создается между блоком 7 и 4.

Процедура создания маркера безопасного доступа. ОС использует идентификатор пользователя для его идентификации, а пароль для подтверждения прав пользователя.

Интерактивный процесс регистрации может быть проиллюстрирован на основе следующей блок-схемы:



A — вход в систему.
B — подсистема безопасности.
C — подсистема аутентификации.
D — SAM.
E — подсистема Win 32.
F — рабочий стол пользователя.

1 — “Ctrl + Alt + Del” — инициирует процесс Logon session и одновременно блокирует доступ к этой сессии любых деструктивных программных средств.

2 — ввод идентификатора и пароля, что должно инициировать работу LSA.

3 — LSA активизирует работу подсистемы или пакета аутентификации.

4 — пакет аутентификации анализирует БД SAM или же если бюджет пользователя касается локальной системы этого компьютера, то введенный пользователем идентификатор и пароль, будут преобразованы, то есть зашифрованы. Если же бюджет не является локальным (нет в БД SAM), то будет запущен альтернативный пакет аутентификации.

5 — после анализа бюджета SAM возвращает ID пользователя или ID всех групп пользователей, к которым относится данный пользователь.

6 — пакет аутентификации создает сеанс входа в систему и инициирует или передает управление сеансу идентификации, то есть передает управление LSA.

7 — если пользователь не получает прав доступа к системе, то процесс регистрации возвращает код ошибки. С другой стороны, если идентификация и аутентификация завершилась успешно, то генерируются маркеры безопасного доступа, которые содержат идентификатор пользователя и групп.

8 — сеанс входа в систему активизирует систему Win 32, чтобы начать сеанс работы с системой, на основе созданного маркера.

9 — подсистема Win 32 создает Desktop данного пользователя.

БД SAM.

Она содержит информацию и идентифицирует и аутентифицирует пользователя. Представляется с помощью 2-ух 16-байтовых двоичных последовательностей на основе алгоритма DES. Как правило, система создает закодированный пароль, анализируя 14 символов пароля. БД SAM является основным объектом атак хакера. На сайте хакеров информация размещается с помощью программы NTBACK.up. Если ее доставить до системы, то ее можно скопировать. Кроме основного файла создается копия SAM. Save, а также сжатая копия, которая создается в файле SAM._. Копия основного файла содержится в подкаталоге …\Config, а сжатая — в подкаталоге …\Repair.

Для того чтобы получить доступ паролей, его необходимо взломать. LOpth Crack и его более новая версия LC3 создают модуль для анализа паролей, пересылают через сеть. Advanced NTSecurity характеризуются очень простым интерфейсом.

Стандартные и наиважнейшие принципы защиты:

1. не использовать стандартных паролей, содержащих имена, даты рождений;
2. длина пароля минимум 8 символов, состоящих на основе минимум 3 групп символов: строчная и прописная буква, цифры, служебные символы (a, A, 1, *);
3. учитывать безопасное время использования пароля;
4. не пользоваться пиратскими программами.

Спам

Спам — нежелательная корреспонденция рекламного или иного характера массово рассылаемая людям, не выразившим желания ее получать.

В первую очередь, термин спам относится к электронным письмам.

Термин «спам» стал использоваться по отношению к электронным сообщениям примерно с 1993 года, когда рекламные компании начали публиковать в группах новостей Usenet, дискуссионных листах, гостевых книгах, сообщения, не имеющие отношения к заданной тематике, или сообщения, являющиеся прямой рекламой.

Первый случай массовой несанкционированной рассылки электронных писем в России был отмечен 19 августа 1991 года: во время августовского путча директор «Релкома» Алексей Солдатов распространил через электронную почту всем своим клиентам обращение Бориса Ельцина[1].

Наиболее распространенные виды спама:

- **Реклама**. Эта разновидность спама встречается наиболее часто — некоторые компании, занимающиеся легальным бизнесом, рекламируют свои товары или услуги с

помощью спама. Они могут осуществлять его рассылку самостоятельно, но чаще заказывают её тем компаниям (или лицам), которые на этом специализируются. Привлекательность такой рекламы заключается в её сравнительно низкой стоимости и (предположительно) большом охвате потенциальных клиентов. В данный момент принимаются законодательные акты против спама.

- **Реклама незаконной продукции.** С помощью спама часто рекламируют продукцию, о которой нельзя сообщить другими способами, например порнографию, лекарственные средства с ограничениями по обороту, ворованную информацию (базы данных), контрафактное программное обеспечение.
- **«Нигерийские письма»** Иногда спам используется для того, чтобы выманить деньги у получателя письма. Такое письмо содержит сообщение о том, что получатель письма может получить каким-либо образом большую сумму денег, а отправитель может ему в этом помочь. Затем отправитель письма просит перевести ему немного денег под предлогом, например, оформления документов или открытия счета. Выманивание этой суммы и является целью мошенников.
- **Фишинг**. Фишинг (англ. phishing от fishing — рыбалка) — ещё один способ мошенничества с помощью спама. Он представляет собой попытку спамеров выманить у получателя письма номера его кредитных карточек или пароли доступа к системам онлайновых платежей. Такое письмо обычно маскируется под официальное сообщение от администрации банка. В нем говорится, что получатель должен подтвердить сведения о себе, иначе его счёт будет заблокирован, и приводится адрес сайта (принадлежащего спамерам) с формой, которую надо заполнить. Среди данных, которые требуется сообщить, присутствуют и те, которые нужны мошенникам. Для того, чтобы жертва не догадалась об обмане, оформление этого сайта также имитирует оформление официального сайта банка.
- **Другие виды спама**

Способы распространения:

- **Электронная почта.** Самый большой поток спама распространяется через электронную почту (e-mail). В настоящее время доля вирусов и спама в общем трафике электронной почты составляет по разным оценкам от 85 до 95 процентов. Спамеры собирают e-mail адреса с помощью специального робота или вручную (редко), используя веб-страницы, конференции Usenet, списки рассылки, электронные доски объявлений, гостевые книги, чаты. Такая программа-робот способна собрать за час тысячи адресов и создать из них базу данных для дальнейшей рассылки

по ним спама. Некоторые компании занимаются только сбором адресов, а базы потом продают.

- **Usenet** Многие группы новостей Usenet, особенно немодерируемые, были покинуты пользователями и сейчас содержат почти исключительно рекламу, часто даже не по теме. Вместо других были созданы модерируемые конференции.
- **Мгновенные сообщения** С развитием служб доставки мгновенных сообщений, таких как ICQ, AIM и др., спамеры стали их использовать для своих целей. Многие из этих служб предоставляют список пользователей, которым можно воспользоваться для рассылки спама.
- **Блоги, Вики** В последнее время стали популярны веб-сайты, которые можно свободно редактировать — блоги и вики. Например, Википедия создается с использованием этой технологии. Так как эти страницы открыты для свободного редактирования, на них может быть размещён спам.
- **SMS-сообщения** Спам может распространяться не только через Интернет. Рекламные сообщения, присылаемые на мобильные телефоны с помощью SMS-сообщений, особенно неприятны тем, что от них труднее защититься, и получатель иногда должен платить за каждое сообщение. Это может быть заметная сумма, особенно если абонент находится в роуминге.

Деструктивные программные средства.

Существует много классификационных методов:

- сканеры (сканеры портов)
- снифферы (анализаторы трафика)
- троянские кони
- вирусы
- программные закладки
- логические бомбы

Сканеры — программное средство для автоматизированного процесса поиска дыр (люков) в защите персонального компьютера, функционирующего на основе протокола TCP/IP.

Начало создания сканеров положено работой по разработке программных средств, которые выполняли важную функцию — поиск дыр в защите. Были написаны специалистами высокого уровня. Со временем эти сканеры стали использовать в корыстных деструктивных целях — для внедрения в ОС. Среди сканеров, классическим является сканер SATAN (Security Administrations Tools for Analyzing Networks). Он был создан специально для Unix.

Jacka — сканер “невидимка”, анализирующий сетевые домены.

Снифферы — анализаторы трафика движения сети. Функционирование снифферов основано на том, что каждый пакет сообщений, пересылаемый в сети, содержит адрес получателя. Сетевой адаптер анализирует, прежде всего, адрес получателя. Если этот адрес является адресом этого компьютера, то анализ закончен. Sniffer — от названия первого продукта — это программа предназначенная для анализа пакетов сообщений, разработана Network General Corporation.

Снифферы сегодня являются коммерческим продуктом. Методы защиты от снифферов — сегментирование локальной системы, использование мостов bridge.

Вирусы

Основы теории самовоспроизводящихся механизмов заложил американец венгерского происхождения Джон фон Нейман (John von Neumann), который в 1951 предложил метод создания таких механизмов. Первой публикацией, посвященной созданию самовоспроизводящихся систем, является статья Л. С. Пенроуз (L. S. Penrose) (жена нобелевского лауреата по физике Р. Пенроуза) о самовоспроизводящихся механических структурах, опубликованная в 1957. американским журналом «Nature». В этой статье, наряду с примерами чисто механических конструкций, была приведена некая двумерная модель подобных структур, способных к активации, захвату и освобождению. По материалам этой статьи Ф. Ж. Шталь (F. G. Stahl) запрограммировал на машинном языке ЭВМ IBM 650 биокибернетическую модель, в которой существа двигались, питаясь ненулевыми словами. При поедании некоторого числа символов существо размножалось, причём, дочерние механизмы могли мутировать. Если кибернетическое существо двигалось определённое время без питания, оно погибало.

Одним из первых определений вируса было определение данное в 1984 году Ф. Коэном(F. Kohen). Вирус — программа, которая может создавать собственные копии, распространять их в других программах и выполнять определенные функции.

В настоящее время не существует единой системы классификации и именования вирусов (хотя попытка создать стандарт была предпринята на встрече CARO в 1991 году). Принято разделять вирусы **по поражаемым объектам** (файловые вирусы, загрузочные вирусы, скриптовые вирусы, сетевые черви), по **поражаемым операционным системам и платформам** (DOS, Windows, Unix, Linux, Java и другие), по **технологиям используемым вирусом** (полиморфные вирусы, стелс-вирусы), по **языку на котором написан вирус** (ассемблер, высокоуровневый язык программирования, скриптовый язык и др.).

Классификация файловых вирусов по способу заражения.

По способу заражения файловые вирусы (вирусы, внедряющие свой код в исполняемые файлы: командные файлы, программы, драйверы, исходный код

программ и др.)разделяют на перезаписывающие, паразитические, вирусы-звенья, вирусы-черви, компаньон-вирусы, а так же вирусы, поражающие исходные тексты программ и компоненты программного обеспечения (VCL, LIB и др.).

Перезаписывающие вирусы. Вирусы данного типа записывают свое тело вместо кода программы, не изменяя названия исполняемого файла, вследствие чего исходная программа перестает запускаться. При запуске программы выполняется код вируса, а не сама программа.

Вирусы-компаньоны. Компаньон-вирусы, как и перезаписывающие вирусы, создают свою копию на месте заражаемой программы, но в отличие от перезаписываемых не уничтожают оригинальный файл, а переименовывают или перемещают его. При запуске программы вначале выполняется код вируса, а затем управление передается оригинальной программе. Возможно существование и других типов вирусов-компаньонов, использующих иные оригинальные идеи или особенности других операционных систем. Например, PATH-компаньоны, которые размещают свои копии в основном катагоге Windows, используя тот факт, что этот каталог является первым в списке PATH, и файлы для запуска Windows в первую очередь будет искать именно в нем. Данными способом самозапуска пользуются также многие компьютерные черви и троянские программы.

Файловые черви. Файловые черви создают собственные копии с привлекательными для пользователя названиями (например Game.exe, install.exe и др.) в надежде на то, что пользователь их запустит.

Вирусы-звенья. Как и компаньон-вирусы, не изменяют код программы, а заставляют операционную систему выполнить собственный код, изменяя адрес местоположения на диске зараженной программы, на собственный адрес. После выполнения кода вируса управление обычно передается вызываемой пользователем программе.

Паразитические вирусы. Паразитические вирусы — это файловые вирусы изменяющие содержимое файла добавляя в него свой код. При этом зараженная программа сохраняет полную или частичную работоспособность. Код может внедряться в начало, середину или конец программы. Код вируса выполняется перед, после или вместе с программой, в зависимости от места внедрения вируса в программу.

Вирусы, поражающие исходный код программ. Вирусы данного типа поражают или исходный код программы, либо её компоненты (OBJ-, LIB-, DCU-файлы) а так же VCL и ActiveX компоненты. После компиляции программы оказываются в неё встроенными. В настоящее время широкого распространения не получили.

Другие виды опасного программного обеспечения.

Троянская программа (троя́н, троя́нец) — разновидность компьютерных программ, которые «претендуют» на то, что выполняют некоторую определенную функцию, в действительности же работают совершенно иначе. Некоторые троянцы изначально спроектированы так, чтобы вводить пользователя в

заблуждение: к примеру, программа имитирует другое приложение (игру, текстовый редактор, программу инсталяции), а на самом деле выполняет некие несанкционированные пользователем действия: удаление, модификацию информации, сбор и пересылку информации третьим лицам, передача управления компьютером удаленному пользователю. Либо в качестве троянца может быть использована штатная программа: некое лицо помещает для скачивания под видом «полезной» программы или обновлений программу форматирования дисков и командный файл, который запускает её с необходимыми параметрами.

Sniffer (от англ. to sniff — нюхать) — сетевой анализатор трафика, программа или программно-аппаратное устройство, предназначенное для перехвата и последующего анализа, либо только анализа сетевого трафика, предназначенного для других узлов.

Перехват трафика может осуществляться:

- обычным «прослушиванием» сетевого интерфейса (метод эффективен при использовании в сегменте концентраторов (хабов) вместо коммутаторов (свичей), в противном случае метод малоэффективен, поскольку на снифер попадают лишь отдельные фреймы);
- подключением снифера в разрыв канала;
- ответвлением (программным или аппаратным) трафика и направлением его копии на снифер;
- через анализ побочных электромагнитных излучений и восстановление таким образом прослушиваемого трафика;
- через атаку на канальном (2) или сетевом (3) уровне, приводящую к перенаправлению трафика жертвы или всего трафика сегмента на снифер с последующим возвращением трафика в надлежащий адрес.

Эксплойт (англ. exploit — использовать) — это общий термин в сообществе компьютерной безопасности для обозначения фрагмента программного кода который, используя возможности предоставляемые ошибкой, отказом или уязвимостью, ведёт к повышению привилегий или отказу в обслуживании компьютерной системы.

Spyware (англ. Spy — шпион и англ. Software — программное обеспечение) — шпионское программное обеспечение. Программное обеспечение, обычно распространяющееся вместе другим полезным, занимающееся сбором информации на компьютере пользователя и отсылкой ее создателю.

Keylogger - это клавиатурный шпион. Программа, которая записывает все нажатия клавиш на компьютере, все пароли, on-line чаты, переписку по электронной почте и так далее. Также программа позволяет записывать все приложения, запускаемые на компьютере, отображать все сайты, куда заходил пользователь, отображать содержимое clipboard, делать скриншоты. Программа позволяет хранить и просматривать записи всех действий на компьютере.

Методы защиты от вредоносного ПО

Для того, чтобы успешно противостоять попыткам вирусов проникнуть на Ваш компьютер необходимо выполнять два простейших условия: соблюдать правила «компьютерной гигиены» и пользоваться антивирусными программами.

Правила “компьютерной гигиены“:

- ни в коем случае не открывайте файлы, присылаемые Вам по электронной почте неизвестными людьми
- осторожно относитесь к файлам, присылаемым Вашими знакомыми и партнерами. Они могут даже и не знать, что с их компьютера вирус незаметно рассылает свои копии людям из их адресной книги!
- обязательно проверяйте антивирусным сканером с максимальным уровнем проверки все дискеты, компакт-диски и другие мобильные носители информации, а также файлы, получаемые из сети Интернет, и других публичных ресурсов (BBS, электронных конференций и т.д.)
- проводите полную антивирусную проверку компьютера после его получения из ремонтных служб. Ремонтники пользуются одними и теми же дискетами для проверки всех компьютеров – они очень легко могут занести «заразу» с другой машины!
- с осторожностью допускайте работать с Вашим компьютером других пользователей
- внимательно следите за предоставляемыми правами доступа к Вашему компьютеру
- своевременно устанавливаете “заплатки“ от производителей используемых Вами операционных систем и программ
- для повышения сохранности Ваших данных периодически проводите резервную архивацию информации на независимые носители

Рефераты

ТУРБО КОДЫ

Введение

Турбо коды представляют собой новый тип кодов для исправления ошибок, возникающих при передаче цифровой информации по каналам связи с шумами. Турбо коды были введены в рассмотрение французским исследователем Claude Berrou в 1993 г. и сразу же привлекли к себе пристальное внимание специалистов в области помехоустойчивого кодирования информации во всем мире. Причина этому - уникальная способность турбо кодов обеспечивать характеристики помехоустойчивости передачи информации по каналам с шумами близкие к теоретически достижимым значениям (так называемый предел Шеннона) при умеренной сложности оборудования для кодирования и декодирования. Уже в первой работе по турбо кодам [C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in ICC'93, Geneva, Switzerland, May 93, pp. 1064-1070] была практически продемонстрирована возможность получения значения средней вероятности ошибки на бит 10-5 для ФМ-2 в канале с аддитивным белым гауссовским шумом при отношении сигнал шум на бит всего 0.7 дБ, что лишь на 0.5 дБ больше теоретического предела для бинарных антиподальных сигналов. Мечта осуществилась - предел Шеннона почти достигнут!

Как устроен кодер турбо кода.

Турбо коды относятся к классу так называемых параллельных каскадных кодов. Принцип построения кодера турбо кода достаточно прост (рис. 1). Из структуры кодера видно, что турбо код представляет собой систематический код в котором проверочная группа образуется из проверочных битов, генерируемых двумя кодерами составных рекурсивных сверточных кодов (РСК), причем информационная последовательность подается в кодер первого РСК (РСК1) непосредственно, а в кодер второго РСК (РСК2) через устройство псевдослучайного перемежения. Схема выкалывания проверочных бит применяется для регулирования общей скорости турбо кода. Причина феноменальной помехоустойчивости турбо кодов лежит в сочетании следующих свойств:

сильная зависимость веса выходной последовательности РСК от вида входной информационной последовательности, т. е. от порядка расположения нулей и единиц в ней;

применение перемежителя для изменения вида входной последовательности, подаваемой на входы кодеров составных РСК.

Сочетание этих свойств приводит к тому, что если при подаче определенной информационной последовательности на вход кодера РСК1 вес его проверочной последовательности оказывается малым, то перемеженная версия этой информационной последовательности, подаваемая на вход кодера РСК2, с

высокой вероятностью приведет к генерации проверочной последовательности большого веса из-за указанного выше свойства РСК. Таким образом, если какая-либо комбинация ошибок не может быть исправлена одним РСК, то это почти наверняка будет сделано с помощью проверочной группы другого РСК и наоборот. Заметьте, что при использовании в составе турбо кода нерекурсивной формы сверточных кодов с такой же корректирующей способностью выигрыш от кодирования оказывается намного меньше! Это происходит как раз потому, что вес выходной последовательности сверточных кодов в нерекурсивной форме слабо зависит от вида входной информационной последовательности.

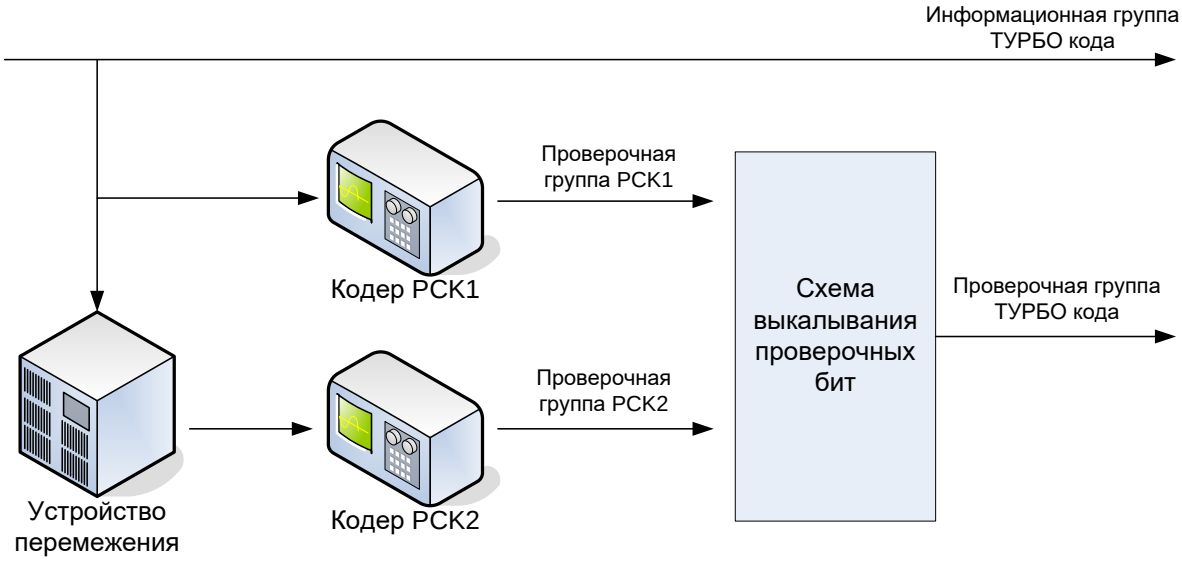


Рис. 1. Кодер классического турбо кода.

Очевидно, что число составных кодов в турбо коде может быть и больше двух. Существуют также схемы построения кодеров с перемежителем в своем составе, которые используют последовательное или гибридное (последовательное и параллельное) соединение кодеров составных кодов, однако приставку турбо в отношении этих конструкций стараются не использовать, несмотря на то, что они обладают столь же высокими рабочими характеристиками. Существуют также турбо коды в которых в качестве составных кодов используются блочные коды, однако такие конструкции несколько уступают классическим турбо кодам по помехоустойчивости и кроме того не позволяют произвольно выбирать длину одновременно кодируемого информационного блока и скорость кода (для сверточных кодов скорость довольно просто реализуется путем применения процедуры выкалывания). Интересно отметить, что после открытия турбо кодов в 1993 году имеется тенденция использовать приставку "турбо" для любой кодовой конструкции хотя бы отдаленно напоминающей турбо код по построению. Яркий пример - так называемые коды произведений, которые теперь пытаются "переименовать" в турбо коды произведений (turbo product codes) мотивируя это тем, что при их декодировании также используется итеративный алгоритм (см. ниже). Однако, эти конструкции были известны задолго до открытия турбо кодов

и не обладают столь высокими характеристиками. Кроме того в составе кодов произведений не используется перемежитель - непременный элемент любого "подлинного" турбо кода. В этой связи не лишне напомнить всем известный лозунг - "Остерегайтесь подделок".

Как устроен декодер турбо кода.

Для декодирования турбо кодов в настоящее время повсеместно применяется концепция так называемого итеративного декодирования сущность которой можно раскрыть, рассматривая структуру итеративного декодера турбо кода (рис. 2).

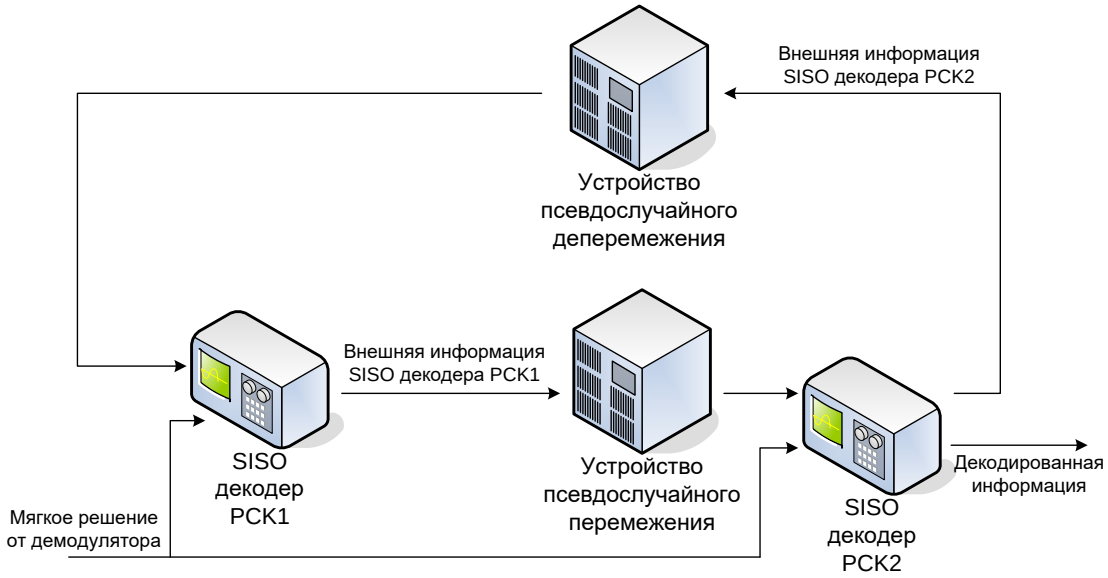


Рис. 2. Структурная схема итеративного декодера турбо кода.

Итеративный декодер образован последовательным соединением декодеров двух элементарных кодов (РСК1 и РСК2), так называемых декодеров с мягкими входным и выходным сигналом (Soft-In Soft-Out - SISO). Каждый элементарный декодер имеет два входа:

вход для сигнала, представляющего собой мягкое решение демодулятора;

вход для сигнала так называемой внешней информации (extrinsic information), получаемой от декодера другого элементарного кода.

Первый декодер в схеме на рис. 2 имеет только один выход, на который подается внешняя информация, полученная этим декодером в процессе декодирования. Внешняя информация, производимая декодером для каждого информационного бита, представляет собой величину, модуль которой пропорционален надежности приема данного информационного бита, а знак соответствует передаче 0 или 1 в данном информационном бите. Существенным

является то, что внешняя информация о каждом информационном бите вырабатывается декодером элементарного кода с использованием сведений об информационных символах, содержащихся только в проверочной группе данного составного кода. Таким образом, внешняя информация оказывается некоррелированной с мягкими решениями, производимыми демодулятором по каждому информационному биту и с информацией о передаваемых информационных символах, содержащейся в проверочной группе другого элементарного кода (разумеется, если мягкие решения, производимые демодулятором по каждому биту, являются статистически независимыми случайными величинами). Поэтому оказывается возможным использовать внешнюю информацию каждого элементарного кода в качестве априорных сведений о передаваемых информационных символах в процессе декодирования в другом элементарном декодере, осуществляя, таким образом, декодирование по методу максимума апостерионой вероятности, что повышает достоверность декодирования. В процессе декодирования турбо кода элементарные SISO декодеры обмениваются друг с другом внешней информацией с каждой итерацией улучшая окончательное решение в смысле снижения средней вероятности ошибки на бит в декодированной информационной последовательности (одна итерация включает в себя декодирование РСК1 и РСК2). Однако, уже после первой итерации внешняя информация, подаваемая на вход декодера РСК1 по цепи обратной связи, оказывается коррелированной с информацией, получаемой из мягких решений демодулятора для проверочных символов РСК1, поэтому улучшение окончательного решения, с каждой итерацией становится меньше и таким образом, величина вероятности ошибки на бит, достигаемая декодированием по этому методу, стремится к определенному пределу. Окончательное (жесткое) решение о передаваемых информационных битах принимается после завершения последней итерации декодером РСК2 и подается на его отдельный выход. Методы построения SISO элементарных декодеров практически сводятся к использованию двух различных алгоритмов декодирования элементарных кодов, способных вырабатывать мягкие выходные решения о передаваемых информационных символах:

BCJR (L. R. Bahl, J. Cocke, F. Jelinek, J. Raviv) алгоритм (другое название - MAP - Maximum Aposteriory Probability) и его упрощенные реализации log-BCJR и max-log-BCJR;

алгоритм Витерби с мягким входным и выходным сигналом (Soft In Soft Out Viterbi Algorithm - SOVA).

BCJR алгоритм позволяет реализовывать характеристики помехоустойчивости турбо кодов, близкие к теоретическим пределам для данных кодов, однако он часто оказывается сложнее в реализации, чем SOVA, который в свою очередь может проигрывать BCJR в помехоустойчивости до 1 дБ при больших длинах блока перемежения и кодового ограничения РСК.

Дополнительную информацию по турбо кодам вы можете найти в pdf

var_Turbo.pdf

файле.

ВЕРНУТЬСЯ К КОНСПЕКТУ

Оглавление

Введение. Сущность проблемы информационной безопасности. Постановка цели и задачи курса.....1
Сущность проблемы информационной безопасности и надежности систем. 3
Характеристика методов и средств защиты информации от несанкционированного доступа. .....**Error! Bookmark not defined.**
Данные. Информация. Их характеристики. Каналы передачи. 13
Характеристика и параметры канала передачи данных.**Error! Bookmark not defined.**
Двоичный канал передачи информации. 15
Базовые методы преобразования информации 19
Цели и задачи преобразования информации. 20
Помехоустойчивое кодирование информации 21
Характеристика надежности передачи данных по каналам симплексного, полудуплексного и дуплексного типов. 22
Временная избыточность 23
Структурная избыточность 25
Структурно временная избыточность 26
Основные понятия теории помехоустойчивого кодирования информации 26
Декодирование кодовых слов. Поиск и исправление ошибок. 32
Практический алгоритм вычисления синдрома 33
Код Хэмминга 33
Код Хэмминга с минимальным кодовым расстоянием $d_{min}=3$ 33
Использование Turbo кодов в системах передачи информации 36
Код Хэмминга с минимальным кодовым расстоянием $d_{min}=4$ 36
Аппаратная реализация кодера и декодера 37
Другие помехоустойчивые коды 39
Составной код. Итерационные коды 39
Модифицированный аддитивный код 39
Характеристика надежности двоичного канала передачи при использовании кодов 41
Суммируя все вышесказанное по первому методу преобразования информации: 42
Преобразование информации на основе методов сжатия (компрессии) 43
Характеристики методов сжатия 43
Символьные методы сжатия 44
Метод Burrows-Wheller 44
Статистические методы сжатия 45
Метод Шеннона-Фано 45
Метод Хаффмана. 49
Словарные методы сжатия данных. 51
Реализация алгоритма Лемпеля-Зива. 55

Принципы построения компрессоров с потерей информации. 60
Криптографические методы преобразования информации. 60
Базовые понятия и определения из области криптографии: 61
Классификация методов криптопреобразования....62
Базовые методы шифрования....63
Характеристика симметричных систем: 65
DES (Data Encryption Standard) 66
Криптографические системы с открытым (публичным) ключом. Ассиметричное шифрование. 73
Электронная цифровая подпись (ЭЦП). 74
Алгоритмы и методы используемые ЭЦП: 74
Характеристика хэш-функции: 78
ЭЦП на основе хэш-функции. 83
Стеганографические методы повышения уровня конфиденциальности информации. 88
Характеристика деструктивных программных средств, методов их внедрения в информационные системы и методов борьбы с ними. 89
Общая характеристика информационных систем с точки зрения их безопасности. 90
RBAC (Role Based Access Control) — управление доступом, основанное на ролях. 91
Защита программного обеспечения. 92
Механизм анализа пароля. 93
Протокол “Керберос” (Protocol Kerberos). 96
Защита регистра. 99
Диспетчер бюджетов безопасности. 100
БД SAM. 103
Спам 103
Деструктивные программные средства. 105
Вирусы 106
Другие виды опасного программного обеспечения. 107
Методы защиты от вредоносного ПО 109
Рефераты 110
ТУРБО КОДЫ 110