Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

В. Г. Спицын

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебное пособие

Томск
«Эль Контент»
2011

УДК 004.43 (031)
ББК 32.973.26-018.2
С72

Рецензенты:
Буймов А. Г., докт. техн. наук, проф., декан экономического факультета ТУСУРа;
Кориков А. М., докт. техн. наук, проф., зав. кафедрой автоматизированных систем управления ТУСУРа.

Спицын В. Г.
С72 Информационная безопасность вычислительной техники : учебное пособие / В. Г. Спицын. — Томск: Эль Контент, 2011. — 148 с.

ISBN 978-5-4332-0020-3

В пособии рассматриваются современные проблемы защиты информации и криптографические методы обеспечения безопасности средств вычислительной техники; определяются основные понятия криптографии и излагаются математические основы криптографических алгоритмов; анализируются наиболее распространенные компьютерные алгоритмы защиты информации; изучаются проблемы антивирусной защиты и создание брандмауэров между локальными и внешними информационно-вычислительных сетями.

УДК 004.43 (031)
ББК 32.973.26-018.2

ISBN 978-5-4332-0020-3

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В предлагаемом вниманию читателя учебном пособии рассматриваются современные проблемы защиты информации, излагаются основные элементы криптографических алгоритмов и обсуждаются методы обеспечения информационной безопасности в информационно-вычислительных сетях.

Первая глава посвящена изложению формулировки проблемы защиты информации и обзору существующих методов защиты информации. Во второй главе содержится описание исторических шифров. В третьей и четвертой главах излагаются основные понятия криптографии и математические основы криптографических алгоритмов. Пятая глава содержит описание наиболее распространенных компьютерных криптографических алгоритмов.

В шестой главе рассматриваются практические вопросы обеспечения компьютерной безопасности на основе применения криптографических алгоритмов при решении задач защиты информации в локальных сетях. В седьмой главе проводится анализ существующих типов вирусов и обсуждаются принципы построения систем антивирусной защиты в локальных и корпоративных сетях. В восьмой главе излагаются существующие конфигурации брандмауэров, структура управления доступом к данным и принципы создания многоуровневой системы безопасности.

Соглашения, принятые в книге

Для улучшения восприятия материала в данной книге используются пиктограммы и специальное выделение важной информации.

Эта пиктограмма означает определение или новое понятие.

Эта пиктограмма означает внимание. Здесь выделена важная информация, требующая акцента на ней. Автор здесь может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.

Эта пиктограмма означает теорему. Данный блок состоит из *Названия теоремы* (Слова Теорема и Номера теоремы) и Текста теоремы.

Пример

Эта пиктограмма означает пример. В данном блоке автор может привести практический пример для пояснения и разбора основных моментов, отраженных в теоретическом материале.

Контрольные вопросы по главе

Глава 1

ПРОБЛЕМЫ И МЕТОДЫ ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

1.1 Информационная безопасность

В настоящее время происходит быстрое развитие средств вычислительной техники и сетей передачи данных, позволяющее на основе оперативного обмена информацией создавать новые направления в науке, банковской, управленческой и предпринимательской деятельности. Проникновение компьютерных технологий во все сферы человеческой жизни привело к необходимости разработки надежных способов решения проблемы защиты информации, циркулирующей в компьютерных системах, от несанкционированного доступа [1–3].

Под информационной безопасностью понимается состояние защищенности обрабатываемых, хранимых и передаваемых в информационно-телекоммуникационных системах (ИТС) данных от незаконного ознакомления, преобразования и уничтожения, а также состояние защищенности информационных ресурсов от воздействий, направленных на нарушение их работоспособности [1, 2].

Основными задачами защиты пользовательской информации являются:

- обеспечение конфиденциальности информации;
- обеспечение целостности информации;
- обеспечение достоверности информации;
- обеспечение оперативности доступа к информации;

- обеспечение юридической значимости информации, представленной в виде электронного документа;
- обеспечение неотслеживаемости действий клиента.

Конфиденциальность — свойство информации быть доступной ограниченному кругу пользователей информационной системы, в которой циркулирует данная информация.

Целостность — свойство информации или программного обеспечения сохранять свою структуру и/или содержание в процессе передачи и/или хранения.

Достоверность — свойство информации, выражающееся в строгой принадлежности объекту, который является ее источником, либо тому объекту, от которого эта информация принята.

Оперативность — способность информации быть доступной для конечного пользователя в соответствии с его временными потребностями.

Юридическая значимость указывает на тот факт, что документ обладает юридической силой. Данное свойство информации особенно актуально в системах электронных платежей.

Неотслеживаемость — способность совершать некоторые действия в информационной системе незаметно для других объектов.

Радикальное решение проблем защиты информации и обеспечения безопасности в ИТС может быть получено только на базе использования криптографических методов и средств защиты информации.

1.2 Проблемы защиты информации в компьютерных системах

Специфическая особенность защиты информации в компьютерных системах заключается в том, что информация не является «жестко связанной» с носителем, а может легко и быстро копироваться и передаваться по каналам связи [1].

Криптографические преобразования данных, основанные на современных скоростных методах, являются наиболее эффективным способом обеспечения конфиденциальности данных, их целостности и подлинности, а в сочетании с необходимыми техническими и организационными мероприятиями могут обеспечить защиту от широкого спектра потенциальных угроз.

Слово криптография в переводе с греческого языка означает «тайнопись».

Криптография исторически появилась в связи с необходимостью передачи секретной информации. Первоначально ее целью являлась разработка специальных методов преобразования информации в форму, недоступную для потенциального злоумышленника. С широкомасштабным применением электронных способов передачи и обработки информации задачи криптографии расширились. Современные проблемы криптографии включают разработку систем электронной цифровой подписи и тайного электронного голосования, идентификации удаленных пользователей, методов защиты от навязывания ложных сообщений и т. д.

В криптографии рассматривается некоторый *нарушитель* (оппонент, криптоаналитик противника, нелегальный пользователь, злоумышленник), который осведомлен об используемых криптографических алгоритмах, протоколах и пытается скомпрометировать их.

Компрометация криптосистемы может заключаться, например, в несанкционированном чтении информации, формировании чужой подписи, не обнаруживаемом модифицировании данных.

Разнообразные действия нарушителя в общем случае называются криптоаналитической атакой.

Специфика криптографии состоит в том, что она направлена на разработку методов, обеспечивающих стойкость к любым действиям злоумышленника [1].

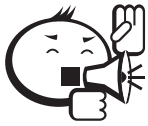
Надежность решения криптографической проблемы (т. е. оценка трудоемкости атаки на криптосистему) является самостоятельным предметом исследования, называемым криптоанализом.

Криптография и криптоанализ составляют единую область знания — криптологию.

Следует отметить, что сертификаты иностранных фирм и организаций в области защиты информации не могут быть заменой отечественным. Сам факт использования зарубежного системного и прикладного программного обеспечения создает повышенную потенциальную угрозу информационным ресурсам.

1.3 Традиционные вопросы криптографии

Классической задачей криптографии является обратимое преобразование некоторого понятного исходного текста (открытого текста) в кажущуюся случайной последовательность некоторых знаков, называемую шифротекстом или криптограммой. При этом шифротекст может содержать как новые, так и имеющиеся в открытом сообщении знаки.



Количество знаков в криптограмме и в исходном тексте в общем случае может различаться. Обязательным требованием является то, что, используя некоторые логические замены символов в шифротексте, можно однозначно и в полном объеме восстановить исходный текст.

Надежность сохранения информации в тайне определялась в прошлые времена тем, что в секрете держался сам метод преобразования [1]. Однако секретность алгоритма не может обеспечить его безусловной стойкости. В связи с этим в настоящее время широко используются открытые алгоритмы, прошедшие длительное тестирование и обсуждение в открытой криптографической литературе.

Стойкость современных криптосистем основывается на секретности некоторой информации сравнительно малого размера, называемой ключом.

Ключ используется для управления процессом криптографического преобразования (шифрования) и является легко сменяемым элементом криптосистемы. Ключ может быть заменен пользователем в произвольный момент времени. В то же время сам алгоритм шифрования является долговременным элементом криптосистемы.

Пробуждение интереса к криптографии, явившееся толчком для ее развития, началось в XIX веке с появлением электросвязи. Наряду с развитием криптографических систем совершенствовались и методы криптоанализа, что приводило к ужесточению требований к криптографическим алгоритмам.

Голландский криптограф Керкхофф (1835–1903) впервые сформулировал изложенное ниже правило [1].

Стойкость шифра (т. е. криптосистемы — набора процедур, управляемых некоторой секретной информацией небольшого объема) должна быть обеспечена в том случае, когда криптоаналитику противника известен весь механизм шифрования за исключением секретного ключа — информации, управляющей процессом криптографических преобразований.

Целью этого требования было осознание необходимости испытания разрабатываемых криптосистем в условиях более жестких по сравнению с теми, в которых мог бы действовать потенциальный нарушитель. Это правило стимулировало появление более качественных шифрующих алгоритмов. Можно сказать, что в нем содержится первый элемент стандартизации в области криптографии. В настоящее время это правило интерпретируется более широко: все долговременные элементы системы защиты считаются известными потенциальному злоумышленнику. В эту формулировку криптосистемы входят как частный случай защиты.

В современном криптоанализе рассматриваются следующие виды нападений на засекречивающие системы [1]:

1) Криптоанализ на основе шифротекста.
2) Криптоанализ на основе известного открытого текста и соответствующего ему шифротекста.
3) Криптоанализ на основе выбранного открытого текста.
4) Криптоанализ на основе выбранного шифротекста.
5) Криптоанализ на основе адаптированного открытого текста.
6) Криптоанализ на основе адаптированного шифротекста.
7) Криптоанализ на основе аппаратных ошибок.

В случае *криптоанализа на основе шифротекста* считается, что противник знает механизм шифрования и ему доступен только шифротекст. Это соответствует модели внешнего нарушителя, который имеет физический доступ к линии связи, но не имеет доступа к аппаратуре шифрования и дешифрирования.

При *криптоанализе на основе открытого текста* предполагается, что криптоаналитику известен шифротекст и та или иная доля исходной информации, а в частных случаях и соответствие между шифротекстом и исходным текстом.

Возможность проведения такой атаки складывается при шифровании стандартных документов, подготавливаемых по стандартным формам, когда определенные блоки данных повторяются и известны.

В нападениях *на основе выбранного открытого текста* предполагается, что криптоаналитик противника может ввести специально подобранный им текст в шифрующее устройство и получить криптограмму, образованную под управлением секретного ключа. Это соответствует модели внутреннего нарушителя.

Криптоанализ на основе выбранного шифротекста предполагает, что противник (оппонент) имеет возможность подставлять для дешифрирования фиктивные шифротексты, которые выбираются специальным образом, чтобы по полученным на выходе дешифратора текстам он мог с минимальной трудоемкостью вычислить ключ шифрования.

Атака *на основе адаптированных текстов* соответствует случаю, когда атакующий многократно подставляет тексты для шифрования (или дешифрирования), причем каждую новую порцию данных выбирает в зависимости от полученного результата преобразований предыдущей порции. Этот вид атаки является наиболее благоприятным для нападающего.

При тестировании новых криптосистем особый интерес представляют нападения на основе известного секретного ключа или ключа шифрования. Следует отметить различия между секретным ключом и ключом шифрования.

Секретный ключ не всегда непосредственно используется для управления процессом шифрования, а часто служит только для выработки расширенного ключа, который используется для шифрования данных.

Ключ, используемый непосредственно для управления процедурами криптографических преобразований, будем называть ключом шифрования.

Очевидно, что существуют шифры, в которых секретный ключ непосредственно используется при шифровании данных, т. е. секретный ключ является одновременно ключом шифрования [1].

1.4 Современные приложения криптографии

Интенсификация информационных потоков в компьютерных сетях, являющаяся результатом все большей автоматизации передачи и обработки данных, существенно расширяет использование криптографических методов не только для обеспечения секретности данных, но и для решения более широкого круга задач. В настоящее время основными направлениями приложений методов криптографии являются [1]:

1) Защита от несанкционированного чтения (или обеспечение конфиденциальности информации).
2) Защита от навязывания ложных сообщений (умышленных и непреднамеренных).
3) Идентификация законных пользователей.
4) Контроль целостности информации.
5) Аутентификация информации.
6) Электронная цифровая подпись.
7) Системы тайного электронного голосования.
8) Защита от отказа факта приема сообщения.
9) Одновременное подписание контракта.
10) Защита документов и ценных бумаг от подделки.

Поскольку первое направление приложений уже рассматривалось, поясним кратко остальные.

Само по себе шифрование данных далеко не всегда является достаточным для *защиты от навязывания ложного сообщения*. В большинстве случаев законный получатель, проанализировав семантику сообщения, может определить, что криптограмма была модифицирована или подменена. Однако при искажении цифровых данных и в некоторых других случаях обнаружить факт искажения данных по семантике крайне сложно. Одним из способов защиты от навязывания ложного сообщения путем целенаправленного или случайного искажения шифротекста является имитозащита [1].

Имитозащита — защита от навязывания ложных сообщений путем формирования в зависимости от секретного ключа специальной дополнительной информации, называемой имитовставкой, которая передается вместе с криптограммой.

Для вычисления имитовставки используется алгоритм, задающий зависимость имитовставки от каждого бита сообщения, причем возможным является любой из 2-х вариантов: (1) вычисление имитовставки осуществляется по открытому тексту и (2) вычисление имитовставки осуществляется по шифротексту. Чем больше длина имитовставки, тем меньше вероятность того, что искажение шифротекста не будет обнаружено законным получателем [1].

Идентификация законных пользователей заключается в распознавании пользователей и основывается на том, что законному пользователю известна информация, недоступная для посторонних. Например, пароль — некоторая случайная секретная информация, сформированная пользователем и не хранящаяся в явном виде в ЭВМ.

Для того чтобы система защиты могла идентифицировать легальных (санкционированных) пользователей, в памяти ЭВМ хранятся образы их паролей, вычисленные по специальному криптографическому алгоритму. Этот алгоритм реализует одностороннюю функцию $y = F(x)$. Основное требование к односторонней функции состоит в том, чтобы сложность вычисления значения функции по аргументу (по входу) была низкой, а сложность определения значения аргумента, для которого найдено значение функции (значение выхода), была высокой (т. е. практически неосуществимой при использовании всех вычислительных ресурсов человечества).

Контроль целостности информации — это обнаружение любых несанкционированных изменений информации, например данных или программ, хранимых в компьютере.

Имитозащита является одним из способов контроля целостности информации, передаваемой в виде шифротекста (частным случаем). Контроль целостности информации, не являющейся секретной и хранящейся в открытом виде (программы, базы данных и т. п.), основан на выработке по некоторому криптографическому закону кода обнаружения модификации (КОМ). КОМ имеет значительно меньший объем, чем охраняемая от модифицирования информация. Основным требованием к алгоритму вычисления КОМ является задание зависимости значения КОМ от каждого двоичного представления всех символов исходного текста [1].

Для обнаружения непреднамеренных искажений данных применяется следующий простой способ.

1) Вычисляется КОМ, соответствующий фиксированному (эталонному) состоянию выбранного блока данных (например, программы).
2) Полученное значение КОМ записывается в таблицу.
3) Перед каждым использованием этого блока данных вычисляется КОМ и сравнивается с соответствующим значением в таблице.

Аутентификация информации — установление санкционированным получателем (приемником) того факта, что полученное сообщение послано санкционированным отправителем (передатчиком).

Электронная цифровая подпись (ЭЦП) основывается на двухключевых криптографических алгоритмах, в которых предусматривается использование 2-х ключей — открытого и секретного. Двухключевые криптоалгоритмы позволяют обеспечить строгую доказательность факта составления того или иного сообщения конкретными абонентами (пользователями) криптосистемы.

Это основано на том, что только отправитель сообщения, который держит в секрете некоторую дополнительную информацию (секретный ключ), может составить сообщение со специфической внутренней структурой. То, что сообщение имеет структуру, сформированную с помощью секретного ключа, проверяется с помощью открытого ключа (процедура проверки ЭЦП) [1–6].

1.5 Понятие криптографического протокола

Протокол — это совокупность действий (инструкций, команд, вычислений, алгоритмов), выполняемых в заданной последовательности двумя или более субъектами с целью достижения определенного результата.

Для того чтобы протокол приводил к желаемой цели, необходимо выполнение следующих условий [1]:

- корректность протокола; совокупность действий, предусмотренных протоколом, должна обеспечить получение требуемого результата при всех возможных ситуациях;
- полнота и однозначность определения, — протокол должен специфицировать действия каждого участника для всех возможных ситуаций;
- непротиворечивость; результаты, полученные различными участниками протокола, не должны быть противоречивыми;
- осведомленность и согласие участников протокола; каждый субъект заранее должен знать протокол и все шаги, которые он должен выполнить; все субъекты должны быть согласны играть свою роль.

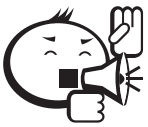
Криптографические протоколы — это такие протоколы, в которых используются криптографические преобразования данных [1].

1.6 Криптография и стеганография

Стеганографией называется техника скрытой передачи или скрытого хранения информации. Целью стеганографии является сокрытие самого факта передачи сообщений [1].

Для этого используются невидимые чернила, невидимые простому глазу пометки у букв, пометки карандашом, плохо заметные отличия в написании букв, микрофотографии и тайники.

В настоящее время стеганографические методы используют распределение по псевдослучайному закону информации в пространстве или времени (применяется также комбинирование этих способов), зашумление и маскирование в некотором сообщении-контейнере или в служебной информации.



Принципиальное отличие криптографии от стеганографии состоит в том, что в ней не скрывается факт передачи сообщений, а скрывается только его содержание (смысл) [1].

Стеганографические методы могут обеспечить высокий уровень защиты информации только в том случае, когда они будут дополнены предварительным, достаточно стойким, криптографическим преобразованием сообщения.

Контрольные вопросы по главе 1

1) Сформулируйте правило Керкхоффа относительно стойкости шифра.
2) Охарактеризуйте криптоанализ на основе шифротекста.
3) В чем заключается криптоанализ на основе известного открытого текста и соответствующего ему шифротекста?
4) Укажите особенности криптоанализа на основе выбранного шифротекста.
5) Опишите криптоанализ на основе адаптированного открытого текста.
6) Каким образом реализуется криптоанализ на основе адаптированного шифротекста.
7) Охарактеризуйте понятие имитозащиты.
8) Каким образом осуществляется контроль целостности информации?
9) Опишите принцип реализации электронной цифровой подписи.
10) Дайте определение криптографического протокола.
11) Дайте определение терминов «криптография» и «стеганография».

Глава 2

ИСТОРИЧЕСКИЕ ШИФРЫ

2.1 Подстановочные и перестановочные шифры

До появления компьютеров криптография состояла из алгоритмов на символьной основе. Различные криптографические алгоритмы либо заменяли одни символы другими, либо переставляли символы. Лучшие алгоритмы делали и то и другое много раз. В настоящее время алгоритмы стали работать с битами, а не с символами, поэтому размер алфавита изменился с 26 элементов до двух. Большинство хороших криптографических алгоритмов до сих пор комбинирует подстановки и перестановки [4–6].

Подстановочным шифром (шифром замены) называется шифр, который каждый символ открытого текста в шифротексте заменяет другим символом. Получатель инвертирует подстановку шифротекста, восстанавливая открытый текст.

В классической криптографии существует четыре типа подстановочных шифров:

1) *Простой подстановочный шифр*, или *моноалфавитный шифр*, — это шифр, который каждый символ открытого текста заменяет соответствующим символом шифротекста.
2) *Однозвучный подстановочный шифр* похож на простую подстановочную криптосистему за исключением того, что один символ открытого текста отображается на несколько символов шифротекста. Например, «A» может соответствовать 5, 13, 25 или 56, «B» — 7, 19, 31 или 42 и так далее.
3) *Полиграмный подстановочный шифр* — это шифр, который блоки символов шифрует по группам. Например, «ABA» может соответствовать «RTQ», «ABB» может соответствовать «SLL» и так далее.

4) *Полиалфавитный подстановочный шифр* состоит из нескольких простых подстановочных шифров. Например, могут быть использованы пять различных простых подстановочных шифров; каждый символ открытого текста заменяется с использованием одного конкретного шифра.

Пример

Примером простого подстановочного шифра является знаменитый *шифр Цезаря,* в котором каждый символ открытого текста заменяется символом, находящимся тремя символами правее по модулю 26 («A» заменяется на «D», «B» — на «E», …, «W» — на «Z», «X» — на «A», «Y» — на «B», «Z» — на «C»). Шифр является простым, так как алфавит шифротекста представляет собой смещенный, а не случайно распределенный алфавит открытого текста.

Простое XOR, представляющее собой операцию «исключающее или» в языке C, является полиалфавитным шифром и широко используется в коммерческих программных продуктах.

Это обычная операция над битами:

$$0 \oplus 0 = 0,$$
$$0 \oplus 1 = 1,$$
$$1 \oplus 0 = 1,$$
$$1 \oplus 1 = 0.$$

У полиалфавитных подстановочных шифров множественные однобуквенные ключи, каждый из которых используется для шифрования одного символа открытого текста. Первым ключом шифруется первый символ открытого текста, вторым ключом — второй символ и так далее. После использования всех ключей они повторяются циклически. Параметр, характеризующий количество ключей, называется *периодом* шифра. В классической криптографии шифры с длинным периодом было труднее раскрыть, чем шифры с коротким периодом. Использование компьютеров позволяет легко раскрыть подстановочные шифры с очень длинным периодом.

Шифр с бегущим ключом (иногда называемый книжным шифром), использующий один текст для шифрования другого текста, представляет собой другой пример подобного шифра. И хотя период этого шифра равен длине текста, он также может быть взломан [3].

Перестановочный шифр — это шифр, в котором меняется не открытый текст, а порядок символов. В простом столбцовом перестановочном шифре открытый текст пишется горизонтально на разграфленном листе бумаги, а шифротекст считывается по вертикали. Дешифрирование представляет собой запись шифротекста вертикально на листе разграфленной бумаги и затем считывание открытого текста горизонтально.

Так как символы шифротекста те же, что и в открытом тексте, частотный анализ шифротекста покажет, что каждая буква встречается приблизительно с той же частотой, что и обычно. Это дает возможность криптоаналитику применить различные методы, определяя правильный порядок символов для получения открытого текста. Применение к шифротексту второго перестановочного шифра значительно повысит безопасность. Существуют и еще более сложные перестановочные шифры, но компьютеры могут раскрыть почти все из них.

В 1920-х годах для автоматизации процесса шифрования были изобретены различные механические устройства. Большинство из них использовало понятие *ротора*, механического колеса, применяемого для выполнения подстановки.

Роторная машина, включающая клавиатуру и набор роторов, реализует вариант шифра Виженера (полиалфавитный подстановочный шифр). Каждый ротор представляет собой произвольное размещение алфавита, имеет 26 позиций и выполняет простую подстановку. Выходные штыри одного ротора соединены с входными штырями следующего ротора.

Например, в четырехроторной машине первый ротор может заменять «A» на «F», второй — «F» на «Y», третий — «Y» на «E» и четвертый — «E» на «C», «C» и будет конечным шифротекстом. Затем некоторые роторы смещаются, и в следующий раз подстановки будут другими. Примером роторной машины является Энигма, применявшаяся Германией в годы второй мировой войны.

2.2 Статистические свойства языка шифрования

В предыдущем разделе рассматривались шифры, применявшиеся для защиты информации в докомпьютерную эпоху.

Установлено, что эти шифры легко взламываются с помощью статистических исследований языка, на котором они написаны (такой язык принято называть подлежащим).

Распределение (встречаемость) букв в английском среднестатистическом тексте представлено в табл. 2.1 [5]. Соответствующая гистограмма изображена на рис. 2.1. Как видно из этих данных, наиболее часто встречающиеся буквы — это «e» и «t».

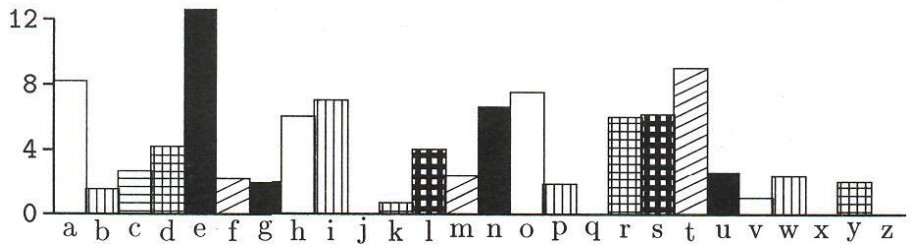


Рис. 2.1 – Относительная встречаемость английских букв

При взломе шифра полезно знать статистическую информацию и второго порядка, а именно частоту встречаемости групп из двух и трех букв, называемых биграммами и триграммами. Наиболее часто встречающиеся биграммы представлены в табл. 2.2, а самые популярные триграммы выписаны в строку в порядке убывания частоты их использования:

the, ing, and, her, ere, ent, tha, nth, was, eth, for.

Таблица 2.1 – Среднестатистические частоты употребления английских букв

Буква	Процентное содержание	Буква	Процентное содержание
a	8,2	n	6,7
b	1,5	o	7,5
c	2,8	p	1,9
d	4,2	q	0,1
e	12,7	r	6,0
f	2,2	s	6,3
g	2,0	t	9,0
h	6,1	u	2,8
i	7,0	v	1,0
j	0,1	w	2,4
k	0,8	x	0,1
l	4,0	y	2,0
m	2,4	z	0,1

Представляет интерес исследовать и взломать несколько классических шифров на основе приведенной статистической информации.

2.3 Шифр сдвига

Коллекцию классических криптосистем открывает один из самых первых известных типов подстановочного шифра, называемый шифром сдвига. Процесс

шифрования заключается в замене каждой буквы на другую, отстоящую от исходной на определенное число позиций в алфавите в зависимости от значения ключа.

Пример

Так, например, если ключ равен 3, его называют шифром Цезаря.

Таблица 2.2 – Частота встречаемости английских биграмм

Биграмма	Процентное содержание	Биграмма	Процентное содержание
Th	3,15	he	2,51
An	1,72	in	1,69
Er	1.54	re	1,48
Es	1.45	on	1.45
Ea	1,31	ti	1,28
At	1,24	st	1,21
En	1,20	nd	1,18

Покажем сейчас, как можно взломать шифр сдвига, опираясь на статистику подлежащего языка. В случае шифра сдвига такой способ не является острой необходимостью. Однако существуют шифры, составленные из нескольких шифров сдвига, которые применяются поочередно, и в этом случае без статистического подхода не обойтись. Кроме того, приложение этого метода к шифру сдвига иллюстрирует проявление статистики исходного открытого текста в соответствующем шифротексте.

Рассмотрим пример шифрограммы [5].

GB OR, BE ABG GB OR: GUNG VF GUR DHRFGVBA:
JURGURE 'GVF ABOYRE VA GUR ZVAQ GB FHSSRE
GUR FYVATF NAQ NEEBJF BS BHGENTRBHF SBEGHAR,
BE GB GNXR NEZF NTNVAFG N FRN BS GEBHOYRF,
NAQ OL BCCBFVAT RAQ GURZ? GB OVR: GB FYRRC;
AB ZBER; NAQ OL N FYRRC GB FNL JR RAQ
GUR URNEG-NPUR NAQ GUR GUBHFNAQ ANGHENY FUBPXF
GUNG SYRFU VF URVE GB, 'GVF N PBAFHZZNGVBA
QRIBHGYL GB OR JVFU'Q, GB QVR, GB FYRRC;
GB FYRRC: CREPUNAPR GB QERNZ: NL, GURER'F GUR EHO;
SBE VA GUNG FYRRC BS QRNGU JUNG QERNZF ZNL PRZR
JURA JR UNIR FUHSSYRQ BSS GUVF ZBEGNY PBVY,
ZHFG TVIR HF CNHFR: GURER'F GUR ERFCRPG
GUNG ZNXRF PNYNZVGL BS FB YBAT YVSR;

Один из приемов взлома этого образца шифротекста основывается на том, что шифровка все еще сохраняет относительные длины слов исходного текста. Например, «N» появляется в нем как однобуквенное слово. Поскольку в английском языке таковыми словами могут быть лишь «a» и «i», легко предположить, что ключ равен либо 13 (т. к. «N» — тринадцатая буква в алфавите после «A»), либо 5 («N» — пятая буква после «I»). Отсюда следует, что пробелы между словами в исходном тексте перед его шифрованием с помощью шифра сдвига следует убирать. Но даже если игнорировать информацию о длине слов, мы можем без особого труда вскрыть шифр сдвига, применив *частотный анализ.*

Вычислим частоты появления букв в шифротексте и сравним их со среднестатистическими из табл. 2.1. Полученная информация представлена на двух гистограммах (рис. 2.2), расположенных друг над другом. Посмотрев на них, легко убедиться, что гистограмма (б), отражающая статистику букв в шифротексте, очень похожа на сдвиг гистограммы (а) со среднестатистической информацией. Заметим, что мы не вычисляли частоты для вычерчивания первой диаграммы по исходному открытому тексту, поскольку он не был нам известен. Мы просто воспользовались легко доступной статистической информацией о подлежащем языке.

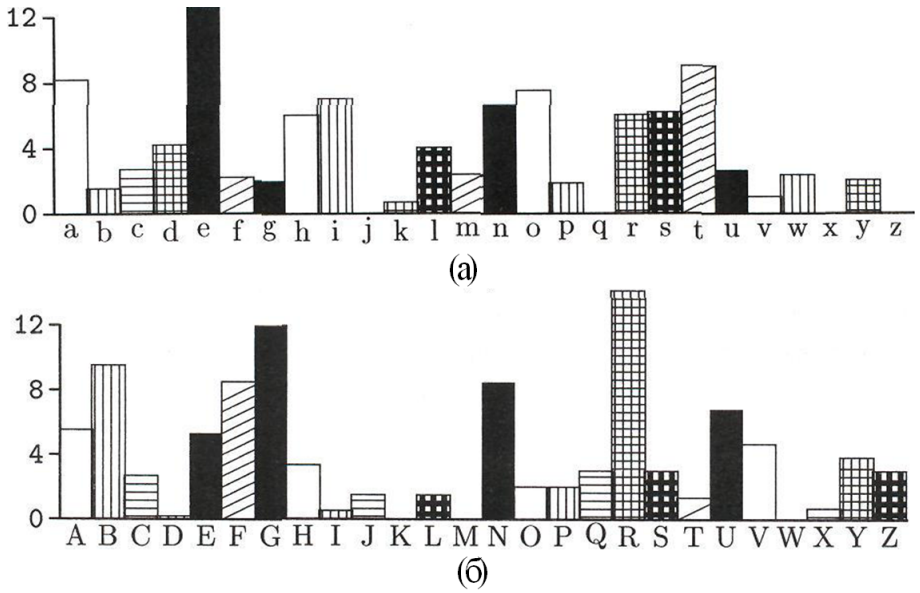


Рис. 2.2 – Сравнение частот появления букв в шифротексте со среднестатистической

Сравнивая гистограммы, можно предположить, насколько вторая сдвинута относительно первой. Наиболее употребляемая буква в английском тексте — это «e». Отмечая большие столбцы во второй гистограмме, замечаем, что буква «e» исходного текста может быть заменена на «G», «N», «R» или «B», т. е. ключом может служить одно из чисел:

2, 9, 13 или 23.

Аналогично, исследуя кандидатуры на букву «a», получаем, что возможный ключ равен одному из

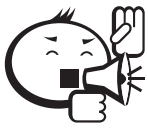
1, 6, 13 или 17.

Среди двух серий гипотетических ключей есть только одно совпадающее значение, а именно 13. Поэтому естественно предположить, что именно 13 и является истинным ключом. Приняв это за рабочую гипотезу, сделаем попытку расшифровать сообщение и обнаружим осмысленный текст.

To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them? To die: to sleep;
No more; and by a sleep to say we end
The heart-ache and the thousand natural shocks
That flesh is heir to, 'tis a consummation
Devoutly to be wish'd. To die, to sleep;
To sleep: perchance to dream: ay, there's the rub;
For in that sleep of death what dreams may come
When we have shuffled off this mortal coil,
Must give us pause: there's the respect
That makes calamity of so long life;

Как видим, здесь приведен знаменитый монолог Гамлета из трагедии Вильяма Шекспира.

2.4 Шифр замены



Основной недостаток шифра сдвига заключается в том, что существует слишком мало возможных ключей, всего 25. В целях устранения указанного недостатка был изобретен *шифр замены.*

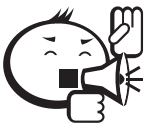
Чтобы описать ключ такого шифра, сначала выписывается алфавит, а непосредственно под ним — тот же алфавит, но с переставленными буквами. Это дает нам правило, по которому буквы открытого текста замещаются символами шифровки [5].

Пример

Например,

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
g	o	y	d	s	i	p	e	l	u	a	v	c	r	j	w	x	z	n	h	b	q	f	t	m	k

Шифрование состоит в замене каждой буквы в открытом тексте на соответствующую ей нижнюю букву. Чтобы расшифровать шифротекст, нужно каждую его букву найти в нижней строке таблицы и заменить ее соответствующей верхней. Таким образом, криптограмма слова *hello* будет выглядеть как *esvvj*, если пользоваться приведенным соответствием.



Количество всех возможных ключей такого шифра совпадает с числом всевозможных перестановок 26 элементов

$$26! \approx 4{,}03 \cdot 10^{26} \approx 2^{88}.$$

Перебирая все возможные ключи с помощью компьютера, необходимо затратить столько времени, что задача о дешифровании конкретного сообщения перестанет быть актуальной. Тем не менее можно взломать шифр замены, опираясь на статистику подлежащего языка, аналогично тому, как было осуществлено вскрытие шифра сдвига.

Рассмотрим пример атаки на криптограмму, в которой оставлены промежутки между словами подлежащего открытого текста.

Рассмотрим шифротекст [5].

XSO MJIWXVL JODIVA STW VAO VY OZJVCO'W
LTJDOWX KVAKOAXJTXIVAW VY SIDS XOKSAVLVDQ IAGZWXJQ.
KVUCZXOJW, KVUUZAIKTXIVAW TAG UIKJVOLOKXJVAIKW
TJO HOLL JOCJOWOAXOG, TLVADWIGO GIDIXTL UOGIT,
KVUCZXOJ DTUOW TAG OLOKXJVAIK KVUUOJKO. TW HOLL TW
SVWXIAD UTAQ JOWOTJKS TAG CJVGZKX GONOLVCUOAX
KOAXJOW VY UTPVJ DLVMTL KVUCTAIOW, XSO JODIVA
STW T JTCIGLQ DJVHIAD AZUMOJ VY IAAVNTXINO AOH
KVUCTAIOW. XSO KVUCZXOJ WKIOAKO GOCTJXUOAX STW
KLVWO JOLTXIVAWSICW HIXS UTAQ VY XSOWO
VJDTAIWTXIVAW NIT KVLLTMVJTXINO CJVPOKXW, WXTYY
WOKVAGUOAXW TAG NIWIXIAD IAGZWXJITL WXTYY. IX STW
JOKOAXLQ IAXJVJZKOG WONOJTL UOKSTAIWUW YVJ
GONOLVCIAD TAG WZCCVJXIAD OAXJOCJOAOZJITL WXZGOAXW
TAG WXTYY, TAG TIUW XV CLTQ T WIDAIYIKTAX JVLO IA
XSO GONOLVCUOAX VY SIDS — XOKSAVLVDQ IAGZWXJQ
I A XSO JODIVA.
XSO GOCTJXUOAX STW T LTJDO CJVDJTUUO VY JOWOTJKS
WZCCVJXOG MQ IAGZWXJQ, XSO OZJVCOTA ZAIVA, TAG
ZE DVNOJAUOAX JOWOTJKS OWXTMLIWSUOAXW TAG

CZMLIK KVJCVJTXIVAW. T EOQ OLOUOAX VY XSIW IW XSO WXJVAD LIAEW XSTX XSO GOCTJXUOAX STW HIXS XSO KVUCZXOJ, KVUUZAIKTXIVAW, UIKJVOLOKXJVAIKW TAG UOGIT IAGZWXJIOW IA XSO MJIWXVL JODIVA. XSO TKTGOUIK JOWOTJKS CJVDJTUUO IW VJDTAIWOG IAXV WONOA DJVZCW, LTADZTDOW TAG TJKSIXOKXZJO, GIDIXTL UOGIT, UVMILO TAG HOTJTMLO KVUCZXIAD, UTKSIAO LOTJAIAD, RZTAXZU KVUCZXIAD, WQWXOU NOJIYIKTXIVA, TAG KJQCXVDJTCSQ TAG IAYVJUTXIVA WOKZJIXQ.

Вычислим частоту встречаемости отдельных букв в этом шифротексте (см. табл. 2.3).

Таблица 2.3 – Частоты встречаемости букв в шифротексте примера

Буква	**Частота**	**Буква**	**Частота**	**Буква**	**Частота**
A	8,6995	B	0,0000	C	3,0493
D	3,1390	E	0,2690	F	0,0000
G	3,6771	H	0,6278	I	7,8923
J	7,0852	K	4,6636	L	3,5874
M	0,8968	N	1,0762	O	11,479
P	0,1793	Q	1,3452	R	0,0896
S	3,5874	T	8,0717	U	4,1255
V	7,2645	W	6,6367	X	8,0717
Y	1,6143	Z	2,7802		

Кроме того, наиболее употребительные биграммы в шифровке — это

$$TA,\ AX,\ IA\ ,\ VA\ ,\ WX\ ,\ XS\ ,\ AG\ ,\ OA\ ,\ JO\ ,\ JV\ ,$$

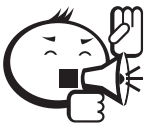
а

$$OAX,\ TAG\ ,\ IVA\ ,\ XSO\ ,\ KVU\ ,\ TXI\ ,\ UOA\ ,\ AXS —$$

чаще всего встречающиеся триграммы.

Поскольку буква «*O*» в нашем образце имеет самую высокую частоту, а именно 11,479, можно предположить, что она соответствует букве «*e*» открытого текста. Посмотрим, что это может означать для наиболее общих триграмм шифротекста.

- Триграмма *OAX* шифротекста соответствует «*e***» исходного сообщения.
- Триграмма *XSO* шифротекста соответствует «***e*» исходного сообщения.



Вспомним теперь часто употребляемые триграммы в английском языке (см. выше) и выберем из них те, которые начинаются или оканчиваются на букву «*e*»: *ent*, *eth* и *the*. Заметим, что первая из популярнейших триграмм шифротекста оканчивается буквой «*X*», а вторая с нее начинается. Аналогичным свойством обладают выбранные триграммы открытого текста: *ent* и *the*. У них есть общая буква «*t*». В связи с этим можно с большой долей вероятности заключить, что имеет место соответствие

$$X=t,\ S=h,\ A=n.$$

Проведенный анализ позволяет облегчить понимание открытого текста, скрытого в шифровке. Ограничившись двумя первыми предложениями, произведем в них замены найденных соответствий, считая, что нашли их правильно.

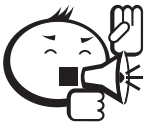
the MJIWtVL JeDIVn haW Vne VY eZJVCe'W LaJDeWt
KVnKentJatIV nW VY hIDh teKhnVLVDQ IndZWtJQ.
KVUCZteJW, KVUUZnIKaTIVnW and UIKJVeLeKtJVnIKW
aJe HeLL JeCJeWented, aLVnDWIde dIDItaL UedIa,
KVUCZteJ, DaUeW and eLeKtJVnIK KVUUeJKe.

Напомним, что такое продвижение в дешифровании произошло после замен:

$$O=e,\ X=t,\ S=h,\ A=n.$$

Теперь мы воспользуемся тем, что в криптограмме оставлены промежутки между словами. Поскольку буква «*T*» появляется в шифровке как отдельное слово, она может замещать лишь одну из двух букв открытого текста: «*i*» или «*a*». Частота буквы «*T*» в шифротексте — 8,0717, а среднестатистические частоты букв «*i*» и «*a*» равны соответственно 7,0 и 8,2 (см. табл. 2.1). Следовательно, скорее всего

$$T=a.$$



Мы уже рассмотрели самую встречаемую триграмму в шифровке, так что обратим наше внимание на следующую по популярности триграмму. Таковой является триграмма *TAG*. Произведя известные замены, увидим, что она означает триграмму *an** открытого текста. Отсюда вполне обоснованно можно сделать вывод: *G=d*, поскольку триграмма *and* — одно из наиболее употребительных буквосочетаний английского языка.

При всех сделанных предположениях о соответствии букв частично дешифрованный кусок шифровки имеет вид:

the MJIWtVL JeDIVn haW Vne VY eZJVCe'W LaJDeWt
KVnKentJatIV nW VY hIDh teKhnVLVDQ IndZWtJQ.
KVUCZteJW, KVUUZnIKatlVnW and UIKJVeLeKtJVnIKW
aJe HeLL JeCJeWented, aLVnDWIde dIDItaL Uedla,
KVUCZteJ, DaUeW and eLeKtJVnIK KVUUeJKe.

Такой результат получился после шести замен:

O=e, X=t, S=h, A=n, T=a, G=d.

На этом этапе исследуем двухбуквенные слова, попадающиеся в криптограмме.

IX. Это слово, как мы знаем, означает **t*. Значит, буква шифра «*I*» может замещать либо «*a*», либо «*i*», т. к. только два двухбуквенных слова английского языка оканчиваются на «*t*»: «*at*» и «*it*». Однако мы уже убедились, что буква «*a*» открытого текста замещается буквой «*T*», так что остается одна возможность: *I=i*.

XV соответствует сочетанию «*t**» открытого текста, откуда *V=o*.

VY можно заменить на «*o**». Поэтому буква «*Y*» шифровки может замещать лишь «*f*», «*n*» или «*r*». Но мы уже знаем букву шифротекста, подменяющую собой «*n*», и у нас остается только две возможности для выбора. Частота встречаемости символа «*Y*» в криптограмме — 1,6, в то время как вероятность встретить букву «*f*» в английском тексте равна 2,2, а букву «*r*» — 6,0. Так что, возможно, имеет место соответствие *Y=f*.

IW должно означать «*i**». Таким образом, «*W*» замещает одну из четырех букв: «*f*», «*n*», «*s*» или «*t*». Так как пары для символов «*f*», «*n*» и «*t*» нам известны, то *W=s*.

Итак, после вычисленных замен:

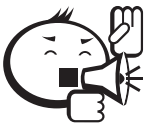
O=e, X=t, S=h, A=n, T=a,
G=d, I=i, V=o, Y=f, W=s

первые два предложения шифротекста выглядят так:

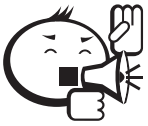
the MJistoL JeDion has one of eZJoCe's LaJDest
KonKentJations of hiDh teKhnoLoDQ indZstJQ.
KoUCZteJs, KoUUZniKations and UiKJoeLeKtJoniKs
aJe HeLL JeCJesented, aLonDside diDitaL Uedia,
KoUCZteJ DaUes and eLeKtJoniK KoUUeJKe.

Даже с половиной определенных букв теперь не очень сложно понять подлежащий открытый текст, взятый с веб-сайта факультета вычислительной математики Бристольского университета. В качестве самостоятельной работы предлагается выявить все оставшиеся буквы и восстановить текст полностью.

2.5 Шифр Виженера



Основной недостаток шифров сдвига и замены заключается в том, что каждая буква открытого текста при шифровании заменяется раз и навсегда фиксированным символом. Поэтому при взломе шифра эффективно работает статистика подлежащего языка.



Шифр замены, описанный в предыдущем разделе, относится к *моноалфавитным* шифрам замены, в которых используется только один упорядоченный набор символов, подменяющий собой стандартный алфавит.

Один из путей решения указанной проблемы состоит в том, чтобы брать несколько наборов символов вместо стандартного алфавита и шифровать буквы открытого текста, выбирая соответствующие знаки из разных наборов в определенной последовательности. Шифры такого типа носят название *полиалфавитных* шифров замены.

Пример

Например, можно рассмотреть такое соответствие [5]:

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
t	m	k	g	o	y	d	s	i	p	e	l	u	a	v	c	r	j	w	x	z	n	h	b	q	f
d	c	b	a	h	g	f	e	m	l	k	j	i	z	y	x	w	v	u	t	s	r	q	p	o	n

в котором первая строка — английский алфавит, а вторая и третья — первый и второй алфавиты шифротекста. В этой ситуации буквы открытого текста, стоящие на нечетных позициях, замещаются соответствующими буквами второй строки, а стоящие на четных — третьей. Таким образом, исходное слово *hello* в шифротексте будет выглядеть как *shljv*. При этом буква «*l*», встречающаяся два раза, замещается разными символами. Если теперь применить частотный анализ, то мы не сможем найти символ шифра, подменяющий собой самую популярную английскую букву «*e*».

На практике можно использовать не два, а вплоть до пяти различных алфавитов шифротекста, многократно увеличивая пространство ключей. Можно подсчитать, что если мы берем символы из пяти замещающих наборов, то число возможных ключей равно $(26!)^5 \approx 2^{441}$.

Однако в этом случае ключ — последовательность из $26 \cdot 5 = 130$ букв. Для того, чтобы усложнить жизнь нарушителю, вскрывающему шифр, необходимо скрыть количество N используемых алфавитов, считая N частью ключа. Для среднего пользователя начала XIX века такая система шифрования представлялась слишком громоздкой, поскольку ключ был слишком большим, чтобы запомнить его. Несмотря на указанный недостаток, самые известные шифры XIX столетия основывались именно на описанном принципе.

Рассмотрим описание шифра Виженера как поточного шифра. Как и в случае шифра сдвига, мы снова перенумеруем буквы. Секретный ключ здесь — это короткая последовательность букв (т. е. слово, часто называемое *лозунгом)*, которая повторяется снова и снова, формируя поток ключей. Кодирование заключается в сложении букв открытого текста с буквами потока ключей (воспринимаемых как числа).

Пример

Например, если ключом является слово *sesame*, то шифрование выглядит так:

```
   t h i s i s a t e s t m e s s a g e
+
   s e s a m e s e s a m e s e s a m e
=
   l l a s u w s x w s f q w w k a s l
```

Заметим, что и в этом примере буква «а» замещается различными символами в зависимости от того, на каком месте открытого текста она стоит.

Шифр Виженера все же не очень сложно взломать, опираясь на статистику подлежащего языка. Как только мы узнаем длину ключевого слова, нам останется несколько раз применить тактику взлома шифра сдвига.

В качестве примера рассмотрим следующую криптограмму:

UTPDHUG NYH USVKCG MVCE FXL KQIB. WX RKU GI TZN,
RLS BHZLXMSNP KDKS; CEB IH HKEW IBA, YYM SRB PFR
SBS, JV UPL O UVADGR HRRWXF. JV ZTVOOV YH ZCQU Y
UKWGEB, PL UQFB P FOUKCG, TBF RQ VHCF R KPG, OU
KFT ZCQU MAW QKKW ZGSY, FP PGM QKFTK UQFB DER EZRN,
MCYE, MG UCTFSVA, WP KFT ZCQU MAW KOIJS. LCOV
NTHDNV JPNUJVB IH GGV RWX ONKCGTHKFL XG VKD, ZJM
VG CCI MVGD JPNUJ, RLS EWVKJT ASGUCS MVGD; DDK
VG NYH PWUV CCHIIY RD DBQN RWTH PFRWBBI VTTK
VCGNTGSF FL IAWU XJDUS, HFP VHSF, RR LAWEY QDFS
RVMEES FZB CHH JRTT MVGZP UBZN FD ATIIYRTK WP KFT
HIVJCI; TBF BLDPWPX RWTH ULAW TG VYCHX KQLJS US

DCGCW OPPUPR, VG KFDNUJK GI JIKKC PL KGCJ IAOV KFTR GJFSAW KTZLZES WG RWXWT VWTL WP XPXGG, CJ FPOS VYC BTZCUW XG ZGJQ PMHTRAIBJG WMGFG. JZQ DPB JVYGM ZCLEWXR: CEB IAOV NYH JIKKC TGCWXF UHF JZK. WX VCU LD YTTKFTK WPKCGVCWIQT PWVY QEBFKKQ, QNH NZTTW IRFL IAS VFRPE ODJRXGSPTC EKWPTGEES, GMCG TTVVPLTFFJ; YCW WV NYH TZYRWH LOKU MU AWO, KFPM VG BLTP VQN RD DSGG AWKWUKKPL KGCJ, XY OPP KPG ONZTT ICUJCHLSF KFT DBQHJTWUG. DYN MVCK ZT MFWCW HTWF FD JL, OPU YAE CH LQ! PGR UF, YH MWPP RXF CDJCGOSE, XMS UZGJQJL, SXVPN HBG!

Существует способ, с помощью которого можно определить длину лозунга, генерирующего поток ключей. Этот способ называют тестом Казисского. Его идея основана на периодичности потока ключей.

Кроме того, в естественном языке существуют часто встречаемые буквосочетания: биграммы и триграммы. Учитывая это, можно предположить, что повторяющиеся наборы символов в шифротексте — след повторений популярных биграмм и триграмм открытого текста. Расстояние между повторениями в таком случае должно быть кратно длине лозунга. Следовательно, вычислив наибольший общий делитель всех таких расстояний, мы получим рабочую гипотезу о длине ключа.

Исследуем расстояния между повторениями биграммы *WX* в приведенном выше шифротексте. Некоторые из расстояний между повторениями этого буквосочетания равны 9, 21, 66 и 30. Отдельные совпадения могут возникнуть случайно, в то время как остальные содержат информацию о длине ключевого слова. Вычислим попарные наибольшие общие делители этих чисел.

НОД (30,66) = 6,
НОД (3,9) = НОД (9,66) = НОД (9,30) = НОД (21,66) = 3.

Маловероятно, что ключевое слово состоит из трех букв, так что будем считать, что расстояния 9 и 21 попали случайно, а длина ключа равна 6.

Возьмем теперь каждую шестую букву шифротекста и применим частотный анализ, как мы уже делали, взламывая шифр сдвига, и определим первую букву ключа.

Преимущество гистограмм в частотном анализе здесь очевидно, поскольку простой перебор всех возможных 26 ключей не сможет дать убедительного результата. Действительно, даже если каждая шестая буква шифротекста будет угадана, получить осмысленный текст мы не сможем.

На рисунке 2.3 дано сравнение частот повторяемости каждой шестой буквы, начиная с первой, со среднестатистической. Изучая эти гистограммы, мы отмечаем

возможные замены букв «a», «e» и «t» естественного языка и делаем вывод, что они сдвинуты на 2. Следовательно, первая буква ключа — «*c*», поскольку именно она обеспечивает такой сдвиг.

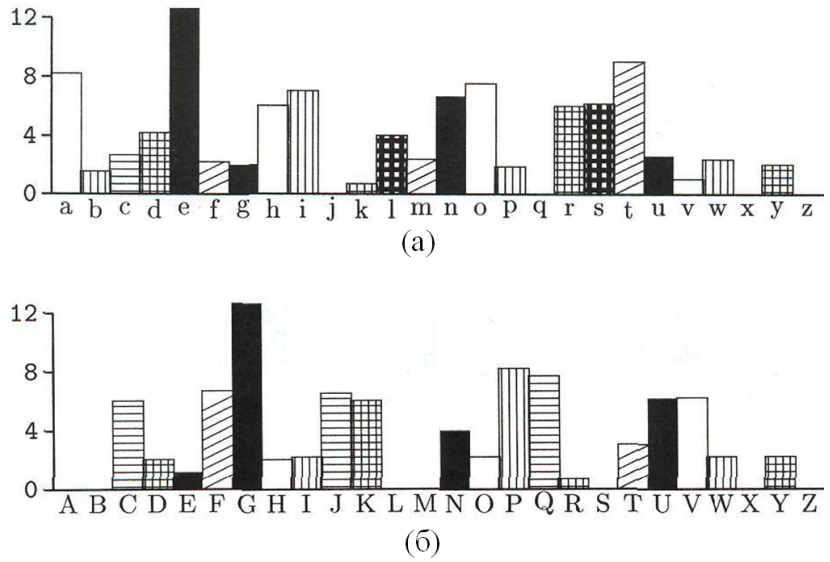


Рис. 2.3 – Сравнение повторяемости каждой шестой буквы, начиная с первой, со среднестатистической (в случае шифра Виженера)

Применим аналогичный анализ для каждой шестой буквы, начиная со второй. Соответствующие гистограммы изображены на рис. 2.4. Используя ту же технику, мы видим, что гистограмма (б) «сдвинута» относительно диаграммы (а) на 17 позиций, что соответствует букве «*r*», стоящей на втором месте ключевого слова.

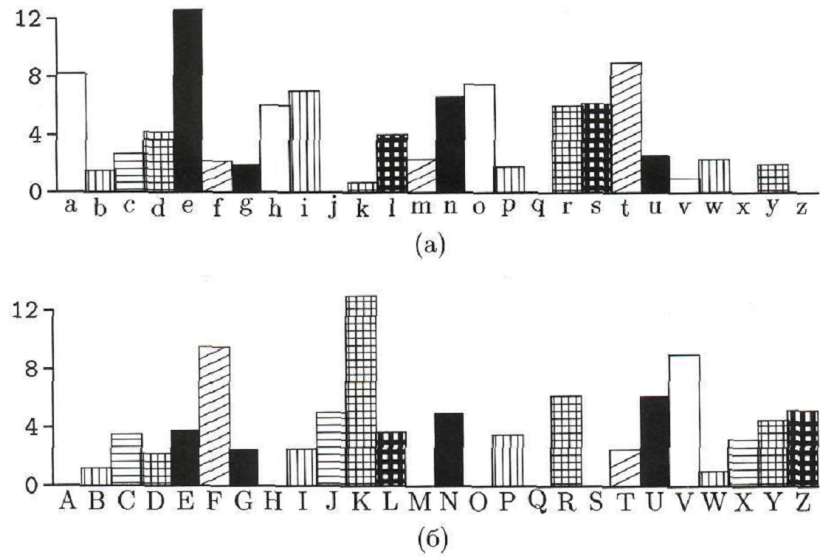


Рис. 2.4 – Сравнение повторяемости каждой шестой буквы, начиная со второй, со среднестатистической

Продолжая поиск, найдем оставшиеся четыре буквы ключа и обнаружим, что лозунг — это «*crypto*». Теперь можно прочесть исходный текст:

Scrooge was better than his word. He did it all, and infinitely more; and to Tiny Tim, who did not die, he was a second father. He became as good a friend, as good a master, and as good a man, as the good old city knew, or any other good old city, town, or borough, in the good old world. Some people laughed to see the alteration in him, but he let them laugh, and little heeded them; for he was wise enough to know that nothing ever happened on this globe, for good, at which some people did not have their fill of laughter in the outset; and knowing that such as these would be blind anyway, he thought it quite as well that they should wrinkle up their eyes in grins, as have the malady in less attractive forms. His own heart laughed: and that was quite enough for him.

He had no further intercourse with Spirits, but lived upon the Total Abstinence Principle, ever afterwards; and it was always said of him, that he knew how to keep Christmas well, if any man alive possessed the knowledge. May that be truly said of us, and all of us! And so, as Tiny Tim observed, God bless Us, Every One!

Приведенный отрывок взят из произведения Чарльза Диккенса «Рождественская песнь в прозе. Святочный рассказ с привидениями».

2.6 Перестановочные шифры

Идеи, лежащие в основе шифра замены, применяются современными криптографами, разрабатывающими симметричные алгоритмы. Например, в шифрах *DES* и *Rijndael* присутствуют компоненты, называемые S-блоками, которые являются простыми подстановками. Другие составляющие современных симметричных шифров основываются на перестановках.

Пример

Предположим, что

$$\sigma = \begin{pmatrix} 1\,2\,3\,4\,5 \\ 2\,4\,1\,3\,5 \end{pmatrix} \in S_5$$

и зашифруем с ее помощью открытый текст:

Once upon a time there was a little girl called Snow White.

Разобьем текст на части по 5 букв:

onceu ponat imeth erewa salit egirl calle dsnow white.

Затем переставим буквы в них в соответствии с нашей перестановкой:

coenu npaot eitmh eewra Isiat iergl lelae ndosw iwthe.

Убрав теперь промежутки между группами, чтобы скрыть значение n, получим шифротекст

coenunpaoteitmheewralsiatiergllclaendoswiwthe.

Перестановочный шифр поддается взлому атакой с выбором открытого текста в предположении, что участвующая в шифровании использованная симметрическая группа (т. е. параметр n) не слишком велика. Для этого нужно навязать отправителю шифрованных сообщений нужный нам открытый текст и получить его в зашифрованном виде. Предположим, например, что нам удалось зашифровать алфавит

abcdefghijklmnopqrstuvwxyz

и получить его шифрованную версию

CADBEHFIGJMKNLORPSQTWUXVYZ.

Сопоставляя открытый текст и криптограмму, получаем перестановку

$$\begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ \ldots \\ 2\ 4\ 1\ 3\ 5\ 7\ 9\ 6\ 8\ 10\ 12\ 14\ 11\ 13\ 15\ \ldots \end{pmatrix}.$$

После короткого анализа полученной информации мы обнаруживаем повторяющиеся серии перестановок: каждые пять чисел переставляются одинаково. Таким образом, можно сделать вывод, что n=5, и восстановить ключ, взяв первые пять столбцов этой таблицы.

2.7 Критерий статистической оценки происхождения шифротекста

При осуществлении криптоанализа перехваченного сообщения криптоаналитику необходимо понять, произошел ли данный шифротекст из чего-либо похожего на естественный язык.

Для проведения соответствующей статистической оценки происхождения шифротекста Фридманом в 1920 г. было предложено применять *индекс совпадения* [5].

Индекс совпадения

Пусть $x_1,\ldots,x_n$ — строка букв, а $f_1,\ldots,f_{25}$ — числа появлений букв в строке. Величина

$$IC(x) = \frac{\sum_{i=0}^{25} f_i (f_i - 1)}{n \cdot (n-1)}$$

называется индексом совпадений.

Заметим, что для случайного набора букв этот индекс равен

$$IC(x) \approx 0{,}038,$$

в то время как для осмысленного текста, написанного на английском или немецком языках, он равен

$$IC(x) \approx 0{,}065.$$

Таким образом, можно сделать следующие выводы:

- Многие шифры, применявшиеся на ранней стадии развития криптологии, были взломаны, поскольку не смогли скрыть статистики подлежащего языка.
- Важнейшие приемы, лежащие в основе первых шифров, — это замены и перестановки.
- Шифры работали с блоками знаков посредством снабженного ключом алгоритма или просто прибавляли очередное число из потока ключей к каждой букве открытого текста.
- Шифры, предназначенные устранить недостатки, связанные со статистикой языка, оказались менее стойкими, чем ожидалось, либо из-за конструкционных недочетов, либо в связи с неграмотной политикой использования.

2.8 Одноразовые блокноты

И все-таки, идеальный способ шифрования существует [6]. Он называется *одноразовым блокнотом* и был изобретен в 1917 году Мэйджором Джозефом Моборном (Major Joseph Mauborgne) и Гилбертом Вернамом (Gilbert Vernam).

В классическом понимании одноразовый блокнот является большой неповторяющейся последовательностью символов ключа, распределенных случайным образом, написанных на кусочках бумаги и приклеенных к листу блокнота.

Первоначально это была одноразовая лента для телетайпов. Отправитель использовал каждый символ ключа блокнота для шифрования только одного символа открытого текста.

Шифрование представляет собой сложение по модулю 26 символа открытого текста и символа ключа из одноразового блокнота. Каждый символ ключа используется только единожды и для единственного сообщения. Отправитель шифрует сообщения и уничтожает использованные страницы блокнота или использованную часть ленты.

Получатель в свою очередь, используя точно такой же блокнот, дешифрирует каждый символ шифротекста. Расшифровав сообщение, получатель уничтожает соответствующие страницы блокнота или часть ленты. Новое сообщение — новые символы ключа [6].

Пример

Например, если сообщением является

ONETIMEPAD,

а ключевая последовательность в блокноте

TBFRGFARFM,

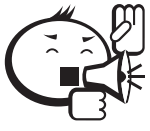
то шифротекст будет выглядеть следующим образом

IPKLPSFHGQ.

Действительно, с учетом того, что алфавит имеет пронумерованный вид, получаем

$$(O + T) \bmod 26 = I, \quad (N + B) \bmod 26 = P, \quad (E + F) \bmod 26 = K,$$
$$(15 + 20) \bmod 26 = 9; \quad (14 + 2) \bmod 26 = 16; \quad (5 + 6) \bmod 26 = 11.$$

В предположении, что злоумышленник не сможет получить доступ к одноразовому блокноту, использованному для шифрования сообщения, эта схема совершенно безопасна. Так как все ключевые последовательности генерируются случайным образом, то у противника отсутствует информация, позволяющая подвергнуть шифротекст криптоанализу.



Случайная ключевая последовательность, сложенная с неслучайным открытым текстом, дает совершенно случайный шифротекст, и никакие вычислительные мощности не смогут его распознать.

Необходимо отметить, что символы ключа должны генерироваться действительно случайным образом. Любые попытки вскрыть такую схему сталкиваются

со способом, которым создается последовательность символов ключа. Использование генераторов псевдослучайных чисел неприемлемо, так как их свойства неслучайны.

Другой важной особенностью использования одноразовых блокнотов является то, что ключевую последовательность никогда нельзя использовать второй раз.

Даже если вы используете блокнот размером в несколько гигабайт, то в случае получения криптоаналитиком ряда текстов с перекрывающимися ключами, он сможет восстановить открытый текст. Для этого достаточно сдвинуть каждую пару шифротекстов относительно друг друга и подсчитать число совпадений в каждой позиции. Если шифротексты смещены правильно, то число совпадений резко возрастет — точное значение зависит от языка открытого текста.

Идея одноразового блокнота легко расширяется на двоичные данные. Вместо одноразового блокнота, состоящего из букв, используется одноразовый блокнот из битов. Применение такого блокнота для шифрования достаточно большого текста связано с рядом проблем. Так как ключевые биты должны быть случайными и не могут использоваться снова, длина ключевой последовательности должна равняться длине сообщения.

Одноразовый блокнот удобен для нескольких небольших сообщений, но его нельзя использовать для работы с каналом связи большой пропускной способности. Даже если проблемы распределения и хранения ключей решены, необходимо точно синхронизировать работу отправителя и получателя. Если получатель пропустит бит (или несколько бит пропадут при передаче), то сообщение потеряет всякий смысл. С другой стороны, если несколько бит изменятся при передаче (и ни один бит не будет удален или добавлен, что гораздо больше похоже на влияние случайного шума), то лишь эти биты будут расшифрованы неправильно. Следует отметить, что одноразовый блокнот не обеспечивает проверку подлинности. Одноразовые блокноты используются и сейчас, главным образом для сверхсекретных каналов связи с низкой пропускной способностью [6].

Контрольные вопросы по главе 2

1) Опишите понятие полиалфавитного подстановочного шифра.
2) Определите понятие книжного шифра.
3) Опишите способ вскрытия шифров на основе применения статистических свойств используемого языка.

4) Охарактеризуйте шифр сдвига.
5) Опишите шифр замены.
6) Охарактеризуйте шифр Виженера.
7) Опишите перестановочный шифр.
8) Каким образом осуществляется шифрование с использованием одноразовых блокнотов?
9) Опишите принцип работы роторной шифровальной машины.
10) Охарактеризуйте критерий статистической оценки происхождения шифротекста.

Глава 3

ОСНОВНЫЕ ПОНЯТИЯ КРИПТОГРАФИИ

3.1 Криптографическая терминология

Отправитель и получатель

Отправитель намерен безопасно послать сообщение получателю, причем он хочет быть уверен, что перехвативший это сообщение не сможет его прочесть.

Сообщение и шифрование

Само сообщение называется открытым текстом (иногда используется термин «клер»). Изменение вида сообщения так, чтобы спрятать его суть, называется шифрованием. Шифрованное сообщение называется шифротекстом. Процесс преобразования шифротекста в открытый текст называется дешифрованием [3, 6]. Эта последовательность показана на рис. 3.1.

Обозначим открытый текст как *M*(от message, т. е. сообщение). Это может быть поток битов, текстовый файл, битовое изображение, оцифрованный звук, цифровое видеоизображение и т. д. Для компьютера *M* — это просто двоичные данные.

Рис. 3.1 – Шифрование и дешифрирование

Обозначим шифротекст как *C* (от ciphertext). Это тоже двоичные данные, иногда того же размера, что и *M*, иногда больше. Если шифрование сопровождается

сжатием, Сможет быть меньше, чем M. Однако само шифрование не обеспечивает сжатие информации.

Функция шифрования E действует на M, создавая C. Соответствующая запись имеет вид:

$$E(M) = C.$$

В обратном процессе функция дешифрирования D действует на C, восстанавливая M:

$$D(C) = M.$$

Поскольку в результате проведения шифрования и последующего дешифрирования исходный текст восстанавливается, то имеет место следующее равенство:

$$D\big(E(M)\big) = M.$$

3.2 Алгоритмы и ключи

Криптографический алгоритм, также называемый шифром, представляет собой математическую функцию, используемую для шифрования и дешифрирования. Обычно это две связанные функции — одна для шифрования, а другая для дешифрования [6].

Современная криптография решает проблемы безопасности с помощью *ключа* K. Такой ключ может быть любым значением, выбранным из большого множества, называемого *пространством ключей*. И шифрование, и дешифрирование определяет этот ключ (т. е. они зависят от ключа, что обозначается индексом K), и теперь эти функции имеют вид:

$$E_K(M) = C,$$
$$D_K(C) = M.$$

При этом выполняется следующее равенство:

$$D_K\big(E_K(M)\big) = C.$$

Для некоторых алгоритмов при шифровании и дешифрировании используются различные ключи, то есть ключ шифрования K_1 отличается от соответствующего ключа дешифрирования K_2. Безопасность этих алгоритмов полностью основана на ключах, а не на деталях алгоритмов.

Криптосистема представляет собой алгоритм, плюс все возможные открытые тексты, шифротексты и ключи [3, 6]. На рис. 3.2 представлена схема шифрования и дешифрирования с ключами.

Существует два основных типа алгоритмов, основанных на ключах, — симметричные и с открытым ключом.

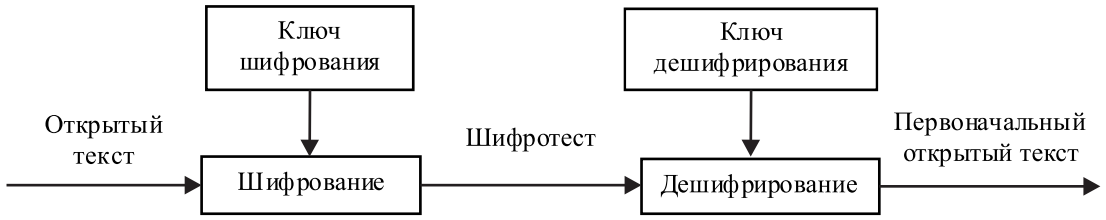


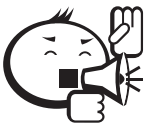
Рис. 3.2 – Шифрование и дешифрирование с ключами

Симметричные алгоритмы представляют собой алгоритмы, в которых ключ шифрования может быть рассчитан по ключу дешифрирования и наоборот.

В большинстве симметричных алгоритмов ключи шифрования и дешифрирования одни и те же. Эти алгоритмы, также называемые алгоритмами с секретным ключом или алгоритмами с одним ключом, требуют, чтобы отправитель и получатель согласовали используемый ключ перед началом безопасной передачи сообщений. Безопасность симметричного алгоритма определяется ключом; раскрытие ключа означает, что кто угодно сможет шифровать и дешифрировать сообщения. Пока передаваемые сообщения должны быть тайными, ключ хранится в секрете [3, 6].

Шифрование и дешифрирование с использованием симметричного алгоритма обозначается как:

$$E_K(M) = C,$$
$$D_K(C) = M.$$



Симметричные алгоритмы делятся на две категории. Одни алгоритмы обрабатывают открытый текст побитно (иногда побайтно), они называются *потоковыми алгоритмами* или *потоковыми шифрами*. Другие алгоритмы работают с группами битов открытого текста. Группы битов называются блоками, а алгоритмы — *блочными алгоритмами* или *блочными шифрами*.

Для алгоритмов, используемых в компьютерных модемах, типичный размер блока составляет 64 бита — достаточно большое значение, чтобы помешать анализу, и достаточно небольшое и удобное — для работы [6].

Алгоритмы с открытым ключом (называемые асимметричными алгоритмами) разработаны таким образом, что ключ, используемый для шифрования, отличается от ключа дешифрирования.

Более того, ключ дешифрирования не может быть (по крайней мере, в течение разумного интервала времени) рассчитан по ключу шифрования. Алгоритмы

называются «с открытым ключом», потому что ключ шифрования может быть открытым: кто угодно может использовать ключ шифрования для шифрования сообщения, но только имеющий соответствующий ключ дешифрирования может расшифровать сообщение. В этих системах ключ шифрования часто называется *открытым* ключом, а ключ дешифрирования — *закрытым*. Закрытый ключ иногда называется секретным ключом [6]. Алгоритмы с открытым ключом были предложены Диффи и Хеллманом в 1975 году. Шифрование с открытым ключом K обозначается как:

$$E_K(M) = C.$$

Хотя открытый и закрытый ключи различны, дешифрирование с соответствующим закрытым ключом обозначается как:

$$D_K(C) = M.$$

Иногда сообщения шифруются закрытым ключом, а дешифрируются открытым, что используется для цифровой подписи.

3.3 Однонаправленные функции

Понятие однонаправленной функции является чрезвычайно важным в криптографии с открытыми ключами. Однонаправленные функции относительно легко вычисляются, но инвертируются с большим трудом. То есть, зная x, просто рассчитать $f(x)$, но по известному $f(x)$ практически невозможно вычислить x [6].

Пример

Хорошим примером однонаправленной функции служит разбитая тарелка. Легко разбить тарелку на тысячу крошечных кусочков, однако нелегко снова сложить тарелку из этих кусочков.

Математически строгого доказательства существования однонаправленных функций нет. Несмотря на это, многие функции выглядят в точности как однонаправленные; мы можем рассчитать их, но до сих пор не знаем простого способа инвертировать их.

Предположим, что однонаправленные функции существуют. Какой смысл останавливаться на них, если непосредственно для шифрования они не используются. Сообщение, зашифрованное однонаправленной функцией, бесполезно — его невозможно дешифрировать.

Пример

Если написать на тарелке что-нибудь, затем разбить ее на крошечные осколки, отдать их получателю, попросив прочитать сообщение, нетрудно догадаться о его реакции на такую однонаправленную функцию.

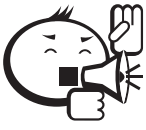
Для криптографии с открытыми ключами необходимо что-то другое, хотя существует и непосредственное криптографическое применение однонаправленных функций.

Однонаправленная функция с люком — это особый тип однонаправленной функции, с секретной лазейкой. Ее легко вычислить в одном направлении и трудно — в обратном. Но если вам известен секрет, вы можете легко рассчитать и обратную функцию. То есть легко вычислить $f(x)$ по заданному x, но трудно по известному $f(x)$ вычислить x. Однако существует небольшая секретная информация y, позволяющая, при знании $f(x)$ и y, легко вычислить x.

Пример

В качестве хорошего примера однонаправленной функции с люком рассмотрим часы. Легко разобрать часы на сотни малюсеньких кусочков, и трудно снова собрать из этих деталей работающие часы. Но с секретной информацией — инструкцией по сборке — намного легче решить эту задачу [6].

3.4 Однонаправленная хэш-функция



У *однонаправленной хэш-функции* может быть множество имен: функция сжатия, функция сокращения (contraction function), краткое изложение, характерный признак, криптографическая контрольная сумма, код целостности сообщения (message integrity check, MIC) и код обнаружения манипуляции (manipulation detection code, MDC). Как бы она ни называлась, эта функция является центральной в современной криптографии.

Хэш-функции, долгое время использующиеся в компьютерных науках, представляют собой функции, которые получают на вход строку переменной длины (называемую прообразом) и преобразуют ее в строку фиксированной, обычно меньшей, длины (называемую значением хэш-функции). В качестве простой хэш-функции можно рассматривать функцию, которая получает прообраз и возвращает байт, представляющий собой XOR всех входных байтов.

Смысл хэш-функции состоит в получении характерного признака, прообраза-значения, по которому анализируются различные прообразы при решении обратной задачи. Так как обычно хэш-функция представляет собой соотношение «многие к одному», невозможно со всей определенностью сказать, что две строки совпадают, но их можно использовать, получая приемлемую оценку точности.

Однонаправленная хэш-функция — это хэш-функция, которая работает только в одном направлении. Легко вычислить значение хэш-функции по прообразу, но трудно создать прообраз, значение хэш-функции которого равно заданной величине.

Хэш-функция является открытой, тайны ее расчета не существует. Безопасность однонаправленной хэш-функции заключается именно в ее однонаправленности. У выхода нет видимой зависимости от входа. Изменение одного бита прообраза приводит к изменению (в среднем) половины битов значения хэш-функции. Вычислительно невозможно найти прообраз, соответствующий заданному значению хэш-функции [6].

С помощью однонаправленных хеш-функций можно получать характерные признаки файлов. В обычных условиях вы можете использовать однонаправленную хэш-функцию без ключа, так что кто угодно может проверить значение хэш-функции. Если нужно, чтобы проверить значение хэш-функции мог только один получатель, необходимо применить код проверки подлинности сообщения.

Код проверки подлинности сообщения (message authentication code, MAC), известный также как код проверки подлинности данных (data authentication code, DAC), представляет собой однонаправленную хэш-функцию с добавлением секретного ключа. Значение хэш-функции является функцией и прообраза, и ключа. Только тот, кто знает ключ, может проверить значение хэш-функции.

3.5 Передача информации с использованием криптографии с открытыми ключами

Пример

Симметричный алгоритм аналогичен сейфу. Ключ является комбинацией. Знающий комбинацию человек может открыть сейф, положить в него документ и снова закрыть. Кто-то другой при помощи той же комбинации может открыть сейф и забрать документ.

В 1976 году Уитфилд Диффи и Мартин Хеллман навсегда изменили эту парадигму криптографии (NSA (National Security Agency) сделало заявление, что знало о такой возможности еще в 1966 году, но доказательств не представило). Они описали *криптографию с открытыми ключами*, используя два различных ключа — один открытый и один закрытый. Определение закрытого ключа по открытому требует огромных вычислительных затрат. Кто угодно, используя открытый ключ, может зашифровать сообщение, но не расшифровать его. Расшифровать сообщение может только владелец закрытого ключа [6].

Пример

Это похоже на превращение криптографического сейфа в почтовый ящик. Шифрование с открытым ключом аналогично опусканию письма в почтовый ящик, любой может сделать это, опустив письмо в прорезь почтового ящика. Дешифрирование с закрытым ключом напоминает извлечение почты из почтового ящика.

Математической основой процесса являются ранее обсуждавшиеся однонаправленные хэш-функции с люком. Шифрование выполняется в прямом направлении. Указания по шифрованию открыты, каждый может зашифровать сообщение. Дешифрирование выполняется в обратном направлении. Оно настолько трудоемко, что, не зная секрета, даже на суперкомпьютерах невозможно расшифровать сообщение за приемлемое время. Секретом, или люком, и служит закрытый ключ, он делает дешифрирование таким же простым, как и шифрование.

Вот как, используя криптографию с открытыми ключами, Алиса может послать сообщение Бобу:

(1) Алиса и Боб согласовывают криптосистему (КС) с открытыми ключами.

(2) Боб посылает Алисе свой открытый ключ.

(3) Алиса шифрует свое сообщение открытым ключом Боба и отправляет его Бобу.

(4) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

Следует обратить внимание на то, что криптография с открытыми ключами устраняет проблему распределения ключей, присущую симметричным криптосистемам. Раньше Алиса и Боб должны были тайно договориться о ключе. Алиса могла выбрать любой ключ, но ей нужно было передать его Бобу. Она могла сделать это заранее, но это требует от нее определенной предусмотрительности. Она могла бы послать ключ с секретным курьером, но для этого нужно время. Криптография с открытыми ключами все упрощает. Алиса может отправить Бобу секретное сообщение без каких-либо предварительных действий. У Евы, подслушивающей абсолютно все, есть открытый ключ Боба и сообщение, зашифрованное этим ключом, но она не сможет получить ни закрытый ключ Боба, ни текст сообщения.

Обычно целая сеть пользователей согласовывает используемую криптосистему. У каждого из них есть открытый и закрытый ключ, открытые ключи помещаются в общедоступной базе данных. Теперь протокол выглядит еще проще:

(1) Алиса извлекает открытый ключ Боба из базы данных.

(2) Алиса шифрует свое сообщение с помощью открытого ключа Боба и посылает его Бобу.

(3) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

В первом протоколе Боб должен был послать Алисе ее открытый ключ прежде, чем она могла отправить ему сообщение. Второй протокол больше похож на обычную почту. Боб не участвует в протоколе до тех пор, пока он не начнет читать сообщение.

3.6 Смешанные криптосистемы

Первые алгоритмы с открытым ключом стали известны в то же время, когда проходило обсуждение DES как предполагаемого стандарта.

В действительности алгоритмы с открытыми ключами не заменяют симметричные алгоритмы и используются не для шифрования сообщений, а для шифрования ключей по следующим двум причинам:

1) Симметричные алгоритмы работают по крайней мере в 1000 раз быстрее, чем алгоритмы с открытыми ключами.

2) Криптосистемы с открытыми ключами уязвимы по отношению к вскрытию с выбранным открытым текстом. Если $C = E(P)$, где P — открытый текст из n-возможных открытых текстов, то криптоаналитику нужно только зашифровать все n-возможные открытые тексты и сравнить результаты с C (помните, ключ шифрования общедоступен). Он не сможет раскрыть ключ дешифрирования, но он сможет определить P.

В большинстве реализаций криптография с открытыми ключами используется для засекречивания и распространения сеансовых ключей, которые используются симметричными алгоритмами для закрытия потока сообщений. Иногда такие реализации называются смешанными (гибридными) криптосистемами.

(1) Боб посылает Алисе свой открытый ключ.

(2) Алиса создает случайный сеансовый ключ, шифрует его с помощью открытого ключа Боба и передает его Бобу

$$E_B(K).$$

(3) Боб расшифровывает сообщение Алисы, используя свой закрытый ключ, для получения сеансового ключа

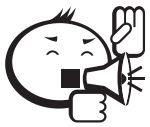
$$D_B\big(E_B(K)\big) = K.$$

(4) Оба участника шифруют свои сообщения с помощью одного сеансового ключа.

Использование криптографии с открытыми ключами для распределения ключей решает очень важную проблему распределения ключей. В симметричной криптографии ключ шифрования данных не используется постоянно. Если злоумышленник получит его, то он сможет расшифровать все закрытые этим ключом сообщения. С помощью приведенного протокола создается сеансовый ключ, который уничтожается по окончании сеанса связи. Это значительно уменьшает риск компрометации сеансового ключа. Конечно, к компрометации чувствителен и закрытый ключ, но риска значительно меньше, так как в течение сеанса этот ключ используется только один раз для шифрования сеансового ключа [6].

3.7 Основные протоколы

Обмен ключами



Общепринятой криптографической техникой является шифрование каждого индивидуального обмена сообщениями отдельным ключом. Такой ключ называется сеансовым, так как он используется для единственного отдельного сеанса обмена информацией. Сеансовые ключи полезны, так как время их существования определяется длительностью сеанса связи.

Передача этого общего сеансового ключа в руки обменивающихся информацией представляет собой сложную проблему.

Протокол, использующий *обмен ключами с помощью симметричной криптографии*, предполагает, что пользователи сети, Алиса и Боб, получают секретный ключ от Центра распределения ключей (Key Distribution Center, KDC) Трента наших протоколов [6]. Перед началом протокола ключи должны быть у пользовате-

лей (протокол игнорирует очень насущную проблему доставки ключей, предполагается, что ключи уже у пользователей, и нарушитель Мэллори не имеет о них никакой информации).

(1) Алиса обращается к Тренту и запрашивает сеансовый ключ для связи с Бобом.

(2) Трент генерирует случайный сеансовый ключ. Он зашифровывает две копии ключа — одну для Алисы, а другую для Боба. Затем Трент посылает обе копии Алисе.

(3) Алиса расшифровывает свою копию сеансового ключа.

(4) Алиса посылает Бобу его копию сеансового ключа.

(5) Боб расшифровывает свою копию сеансового ключа.

(6) Алиса и Боб используют этот сеансовый ключ для безопасности обмена информацией.

Этот протокол основан на абсолютной надежности Трента, для роли которого больше подходит заслуживающая доверия компьютерная программа, чем заслуживающий доверия человек. Большой проблемой является то, что Трент должен участвовать в каждом обмене ключами, и если с ним что-то случится, то это разрушит всю систему.

Обмен ключами с использованием криптографии с открытыми ключами. Ранее нами обсуждалась смешанная КС. Для согласования сеансового ключа Алиса и Боб применяют криптографию с открытыми ключами, а затем используют этот сеансовый ключ для шифрования данных. В ряде реализаций подписанные ключи Алисы и Боба доступны в некоторой базе данных. Это облегчает протокол; теперь Алиса, даже если Боб о ней никогда не слышал, может послать Бобу сообщение.

(1) Алиса получает открытый ключ Боба из KDC.

(2) Алиса генерирует случайный сеансовый ключ, зашифровывает его открытым ключом Боба и посылает его Бобу.

(3) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

(4) Алиса и Боб шифруют свой обмен информацией этим сеансовым ключом.

Вскрытие «человек-в-середине»

В то время как Ева не может сделать ничего лучшего, чем пытаться взломать алгоритм с открытыми ключами или выполнить вскрытие с использованием только шифротекста, у нарушителя Мэллори гораздо больше возможностей [6]. Он не только может подслушать сообщения Алисы и Боба, но и изменить сообщения, удалить сообщения и создать совершенно новые. Мэллори может выдать себя за Боба, сообщающего что-то Алисе, или за Алису, сообщающую что-то Бобу.

Вот как будет выполнено вскрытие.

(1) Алиса посылает Бобу свой открытый ключ. Мэллори перехватывает его и посылает Бобу свой собственный открытый ключ.

(2) Боб посылает Алисе свой открытый ключ. Мэллори перехватывает его и посылает Алисе, как и Бобу, свой собственный открытый ключ.

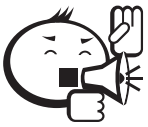
(3) Когда Алиса посылает сообщение Бобу, зашифрованное открытым ключом «Боба», Мэллори перехватывает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он расшифровывает его своим закрытым ключом, затем снова зашифровывает открытым ключом Боба и посылает Бобу.

(4) Когда Боб посылает сообщение Алисе, зашифрованное открытым ключом «Алисы», Мэллори перехватывает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он расшифровывает его, снова зашифровывает открытым ключом Алисы и посылает Алисе.

Это вскрытие будет работать, даже если открытые ключи Алисы и Боба хранятся в базе данных. Мэллори может перехватить запрос Алисы к базе данных и подменить открытый ключ Боба своим собственным. То же самое он может сделать и с открытым ключом Алисы. Или, еще лучше, он может взломать базу данных и подменить открытые ключи Боба и Алисы своим. Затем он дожидается момента, когда Алиса и Боб начнут обмениваться сообщениями, и начинает перехватывать и изменять эти сообщения.

Такое *вскрытие «человек-в-середине»* работает, если у Алисы и Боба нет способа проверить, действительно ли они общаются именно друг с другом.

Протокол «держась за руки»



Протокол «держась за руки», изобретенный Роном Ривестом (Ron Rivest) и Эди Шамиром (Adi Shamir), предоставляет неплохую возможность избежать вскрытия «человек-в-середине» [6].

Вот как он работает.

(1) Алиса посылает Бобу свой открытый ключ.

(2) Боб посылает Алисе свой открытый ключ.

(3) Алиса зашифровывает свое сообщение открытым ключом Боба. Половину зашифрованного сообщения она отправляет Бобу.

(4) Боб зашифровывает свое сообщение открытым ключом Алисы. Половину зашифрованного сообщения он отправляет Алисе.

(5) Алиса отправляет Бобу вторую половину зашифрованного сообщения.

(6) Боб складывает две части сообщения Алисы и расшифровывает его с помощью своего закрытого ключа. Боб отправляет Алисе вторую половину своего зашифрованного сообщения.

(7) Алиса складывает две части сообщения Боба и расшифровывает его с помощью своего закрытого ключа.

Идея в том, что половина зашифрованного сообщения бесполезна без второй половины, она не может быть дешифрирована. Боб не сможет прочитать ни одной части сообщения Алисы до этапа (6), а Алиса не сможет прочитать ни одной части

сообщения Боба до этапа (7). Существует множество способов разбить сообщение на части, например:

- если используется блочный алгоритм шифрования, половина каждого блока (например, каждый второй бит) может быть передана во второй части сообщения;
- первая половина сообщения может быть однонаправленной хэш-функцией шифрованного сообщения, а вторая половина — собственно шифрованным сообщением.

Чтобы понять, как такой протокол помешает Мэллори, рассмотрим его попытку нарушить протокол. Как и раньше, он может подменить открытые ключи Алисы и Боба своим ключом на этапах (1) и (2). Но теперь, перехватив половину сообщения Алисы на этапе (3), он не сможет расшифровать ее своим закрытым ключом и снова зашифровать открытым ключом Боба. Он может создать совершенно новое сообщение и отправить половину его Бобу. Перехватив половину сообщения Боба Алисе на этапе (4), Мэллори столкнется с этой же проблемой. Он не сможет расшифровать ее своим закрытым ключом и снова зашифровать открытым ключом Алисы. Ему придется создать совершенно новое сообщение и отправить половину его Алисе. К тому времени, когда он перехватит вторые половины настоящих сообщений на этапах (5) и (6), подменять созданные им новые сообщения будет слишком поздно. Обмен данными между Алисой и Бобом изменится радикально.

Следует отметить, что использование цифровой подписи в протоколе обмена сеансовым ключом также позволяет избежать вскрытия «человек-в-середине».

Передача ключей и сообщений

Алисе и Бобу не обязательно выполнять протокол обмена ключами перед обменом сообщениями. В этом протоколе Алиса отправляет Бобу сообщение без предварительного протокола обмена ключами [6].

(1) Алиса генерирует случайный сеансовый ключ K и зашифровывает M этим ключом

$$E_K(M).$$

(2) Алиса получает открытый ключ Боба из базы данных.

(3) Алиса шифрует K открытым ключом Боба

$$E_B(K).$$

(4) Алиса посылает Бобу шифрованное сообщение и сеансовый ключ

$$E_K(M),\ E_B(K).$$

Для дополнительной защиты от вскрытия «человек-в-середине» Алиса подписывает передачу.

(5) Боб расшифровывает сеансовый ключ Алисы, K, используя свой закрытый ключ.

(6) Боб, используя сеансовый ключ, расшифровывает сообщение Алисы.

Подобная смешанная система и употребляется чаще всего в системах связи. Ее можно соединить с цифровыми подписями, метками времени и другими протоколами обеспечения безопасности.

Подпись документа с помощью криптографии с открытыми ключами

Существуют алгоритмы с открытыми ключами, которые можно использовать для цифровых подписей. В некоторых алгоритмах — примером является RSA (см. раздел 5.11) — для шифрования может быть использован или открытый, или закрытый ключ. Зашифруйте документ своим закрытым ключом, и вы получите надежную цифровую подпись [6].

В других случаях — примером является DSA — для цифровых подписей используется отдельный алгоритм, который невозможно использовать для шифрования. Эта идея впервые была предложена Диффи и Хеллманом.

Такая подпись соответствует всем требованиям:

- Эта подпись достоверна. Когда Боб расшифровывает сообщение от Алисы с помощью ее открытого ключа, он знает, что именно Алиса подписала это сообщение.
- Эта подпись неподдельна. Только Алиса знает свой закрытый ключ.
- Эту подпись нельзя использовать повторно. Подпись является функцией документа и не может быть перенесена на другой документ.
- Подписанный документ нельзя изменить. После любого изменения документа подпись не сможет больше подтверждаться открытым ключом Алисы.
- От подписи невозможно отказаться. Бобу не требуется помощь Алисы при проверке ее подписи.

Подпись документа и метки времени

На самом деле при определенных условиях Боб сможет смошенничать. Он может повторно использовать документ и подпись совместно. Это не имеет значения, если Алиса подписала контракт (одной копией подписанного контракта больше, одной меньше), но что если Алиса поставила цифровую подпись под чеком?

Предположим, что Алиса послала Бобу подписанный чек на 100. Боб отнес чек в банк, который проверил подпись и перевел деньги с одного счета на другой. Боб сохранил копию электронного чека. На следующей неделе он снова отнес его в этот или другой банк. Банк подтвердил подпись и перевел деньги с одного счета на другой. Если Алиса не проверяет свою чековую книжку, то Боб сможет проделывать это годами.

Поэтому в цифровые подписи часто включают метки времени. Дата и время подписания документа добавляются к документу и подписываются вместе со всем содержанием сообщения. Банк сохраняет эту метку времени в базе данных. Теперь, если Боб попытается получить наличные по чеку Алисы во второй раз, банк проверит метку времени по своей базе данных и откажет Бобу [6].

Стюарт Хабер (Stuart Haber) и В. Скотт Старнетта (W. Scott Starnetta) обеспечили безопасность протоколов связи на основе реализации цифровых меток времени со следующими свойствами:

- *метка времени должна существовать сама по себе, вне зависимости от физической среды, используемой для ее хранения;*
- *не должно существовать возможности изменить хотя бы один бит документа;*
- *не должно существовать возможности задать для документа метку времени, отличного от текущего времени.*

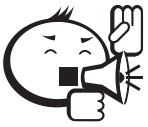
Контрольные вопросы по главе 3

1) Дайте определение однонаправленной функции.
2) Охарактеризуйте понятие симметричного шифра.
3) Опишите принцип работы асимметричных (двухключевых) криптоалгоритмов.
4) Каким образом можно определить понятие однонаправленной хэш-функции?
5) Охарактеризуйте смешанные криптосистемы.
6) Как осуществляется вскрытие «человек-в-середине»?
7) Опишите протокол «держась за руки».
8) Каким образом осуществляется передача ключей и сообщений без предварительного выполнения протокола обмена ключами?
9) Опишите способ подписи документа на основе криптографии с открытыми ключами.
10) Опишите свойства меток времени в электронных цифровых подписях документов?

Глава 4

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИЧЕСКИХ МЕТОДОВ

4.1 Теория информации



Современная теория информации впервые была опубликована в 1948 году Клодом Э. Шенноном (Claude Elmwood Shannon) [7].

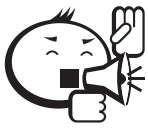
Рассмотрим основные понятия и определения [6, 7].

Энтропия и неопределенность

Теория информации определяет количество информации в сообщении как минимальное количество бит, необходимое для кодирования всех возможных значений сообщения, считая все сообщения равновероятными.

Пример

Например, для поля *дня* недели в базе данных достаточно использовать три бита информации, так как вся информация может быть закодирована тремя битами: 000 — воскресенье, 001 — понедельник, 010 — вторник, 011 — среда, 100 — четверг, 101 — пятница, 110 — суббота, 111 — не используется.



Формально количество информации в сообщении M измеряется *энтропией* сообщения $H(M)$.

Энтропия сообщения, определяющего день недели, немного меньше, чем три бита. В общем случае энтропия сообщения, измеряемая в битах, равна $\log_2 n$, где n — это количество возможных значений:

$$H(M) = \log_2 n.$$

При этом предполагается, что все значения равновероятны. Энтропия сообщения также является мерой его *неопределенности*. Это количество битов открытого текста, которое нужно раскрыть в шифротексте сообщения, чтобы узнать весь открытый текст.

Норма языка

Для данного языка норма языка равна:

$$r = \frac{H(M)}{N},$$

где N — это длина сообщения.

При больших N норма обычного английского языка принимает различные значения от 1,0 бит/буква до 1,5 бит/буква. Шеннон утверждал, что энтропия зависит от длины текста. Он показал, что норма для 8-буквенных блоков равна 2,3 бит/буква, но ее значение падает и находится между 1,3 и 1,5 для 16-буквенных блоков. Томас Кавер (Thomas Cover) использовал игровую методику оценки и обнаружил, что энтропия равна 1,3 бит/символ. В дальнейшем мы также будем использовать это значение.

Абсолютная норма языка равна максимальному количеству битов, которое может быть передано каждым символом при условии, что все последовательности символов равновероятны. Если в языке L символов, то абсолютная норма равна:

$$R = \log_2 L.$$

Это максимум энтропии отдельных символов. Для английского языка с 26 буквами абсолютная норма равна $\log_2 26$, или около 4,7 бит/буква. Следует отметить, что действительная норма английского языка намного меньше, чем абсолютная, так как естественные языки обладают высокой избыточностью.

Избыточность языка, обозначаемая D, определяется как:

$$D = R - r,$$

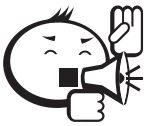
где R — абсолютная норма языка, r — норма языка.

Считая, что норма английского языка равна 1,3, избыточность составит 3,4 бит/буква. Это означает, что каждая английская буква содержит 3,4 бита избыточной информации.

У сообщения ASCII, состоящего только из английских букв, количество информации на каждый байт составляет 1,3 бита. Значит, в каждом байте содержится 6,7 (8-1,3) бита избыточной информации, что дает общую избыточность 0,84 (6,7/8=0,84) бита информации на бит ASCII-текста и энтропию 0,16 (1-0,84=0,16) бита информации на бит ASCII-текста.

Безопасность криптосистемы

Шеннон определил точную математическую модель понятия безопасности криптосистемы. Существуют криптосистемы, достигающие *совершенной безопасности*. Такой является криптосистема, в которой шифротекст не дает никакой информации об открытом тексте (кроме, возможно, его длины). Шеннон теоретически показал, что такое возможно, только если число возможных ключей так же велико, как и число возможных сообщений. Другими словами, ключ должен быть не короче самого сообщения и не может использоваться повторно.



Это означает, что единственной системой, которая достигает идеальной безопасности, может быть только криптосистема с одноразовым блокнотом.

За исключением идеально безопасных систем, шифротекст неизбежно дает определенную информацию о соответствующем открытом тексте. Хороший криптографический алгоритм сохраняет минимум этой информации. Криптоаналитики используют естественную избыточность языка для уменьшения числа возможных открытых текстов.

Энтропия криптосистемы является мерой размера пространства ключей, K. Она приблизительно равна логарифму числа ключей по основанию 2:

$$H(K) = \log_2 K.$$

Энтропия криптосистемы с 64-битовым ключом равна 64 битам, энтропия криптосистемы с 56-битовым ключом равна 56 битам. Чем больше энтропия, тем труднее взломать криптосистему.

Расстояние уникальности

Шеннон определил расстояние уникальности, U, называемое также точкой уникальности, как такое приближенное количество шифротекста, для которого сумма реальной информации (энтропия) в соответствующем открытом тексте $H(M)$ плюс энтропия ключа шифрования $H(K)$ равняется числу используемых битов шифротекста [6]:

$$H(M) + H(K) = n.$$

Он также показал, что шифротексты, которые длиннее расстояния уникальности, можно расшифровать, вероятнее всего, только одним осмысленным способом. Что касается шифротекстов, которые заметно короче расстояния уникальности, их можно расшифровать, скорее всего, несколькими способами, каждый из которых может быть правилен, и таким образом поставить перед противником задачу выбора правильного открытого текста [6, 7].

Для большинства симметричных криптосистем расстояние уникальности определяется как энтропия криптосистемы, деленная на избыточность языка:

$$U = \frac{H(K)}{D}.$$

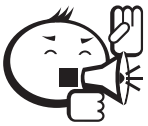
Расстояние уникальности является не точным, а вероятностным значением. Оно позволяет оценить минимальное количество шифротекста, при вскрытии которого методом перебора имеется, вероятно, только один разумный способ дешифрирования. Обычно, чем больше расстояние уникальности, тем лучше криптосистема.

Для DES с 56-битовым ключом и англоязычного сообщения, записанного символами ASCII, расстояние уникальности приблизительно равно 8,2 символа ASCII или 66 бит. Расстояние уникальности измеряет количество криптотекста, необходимое для единственности результата криптоанализа. Расстояние уникальности обратно пропорционально избыточности.

Шеннон определил КС с бесконечным расстоянием уникальности, как обладающую *идеальной тайной*. Если криптосистема обладает идеальной тайной, то даже при успешном криптоанализе останется некоторая неопределенность, является ли восстановленный открытый текст реальным открытым текстом [6, 7].

Практическое использование теории информации

Хотя эти понятия имеют большое теоретическое значение, реальный криптоанализ использует их достаточно редко. Расстояние уникальности гарантирует ненадежность системы, если оно слишком мало, но его высокое значение не гарантирует безопасности. Криптоаналитики используют ряд тестов на базе статистики и теории информации, чтобы выбирать наиболее перспективные направления анализа.



Следует отметить, что большинство литературы по применению теории информации в криптоанализе остается секретной, включая основополагающую работу Алана Тьюринга (Alan Turing), написанную в 1940 г. [6].

4.2 Теория сложности

Теория сложности обеспечивает методологию анализа *вычислительной сложности* различных криптографических методов и алгоритмов. Она сравнивает криптографические методы и алгоритмы и определяет их безопасность. Теория информации сообщает нам о том, что все криптографические алгоритмы (кроме одноразовых блокнотов) могут быть взломаны. Теория сложности сообщает, могут ли они быть взломаны за приемлемое время [6].

Сложность алгоритмов

Сложность алгоритма определяется вычислительными мощностями, необходимыми для его выполнения. Вычислительная сложность алгоритма часто измеряется двумя параметрами: T (временная сложность) и S (пространственная сложность, или требования к памяти). И T, и S обычно представляются в виде функций от n, где n — это размер входных данных.

Обычно вычислительная сложность алгоритма выражается с помощью нотации «О большого», т. е. описывается порядком величины вычислительной сложности. Это просто член разложения функции сложности, быстрее всего растущий с ростом n, все члены низшего порядка игнорируются. Например, если временная сложность данного алгоритма равна $4n^2 + 7n + 12$, то вычислительная сложность составляет величину порядка n^2 и записывается как $O(n^2)$.

Временная сложность, измеренная таким образом, не зависит от реализации. При этом не нужно знать точное время выполнения различных операций. Эта нотация позволяет увидеть, как объем входных данных n влияет на требования ко времени и к объему памяти. Например, если $T = O(n)$, то удвоение входных данных удвоит и время выполнения алгоритма. Если $T = O(2^n)$, то добавление одного бита к входным данным удвоит время выполнения алгоритма.

Алгоритмы классифицируются в соответствии с их временной или пространственной сложностью. Алгоритм называют *постоянным*, если его сложность не зависит от n: $O(1)$. Алгоритм является *линейным*, если его временная сложность $O(n)$. Алгоритмы могут быть *квадратичными*, *кубическими* и т. д. Все эти алгоритмы — *полиномиальны*, их сложность — $O(n^m)$, где m — константа. Алгоритмы с полиномиальной временной сложностью называются алгоритмами *с полиномиальным временем*.

Алгоритмы, сложность которых равна $O(t^{f(n)})$, где t — константа, большая, чем 1, а $f(n)$ — некоторая полиномиальная функция от n, называются *экспоненциальными*.

С ростом n временная сложность алгоритмов может стать настолько огромной, что это повлияет на практическую реализуемость алгоритма. В табл. 4.1 показано время выполнения для различных классов алгоритмов при n, равном одному миллиону. В таблице игнорируются постоянные величины, но показано, почему это можно делать.

Таблица 4.1 – Время выполнения для различных классов алгоритмов

Класс	**Сложность**	**Количество операций для** $n = 10^6$	**Время при** 10^6 **операций в секунду**
Постоянные	$O(1)$	1	1 мкс
Линейные	$O(n)$	10^6	1 с
Квадратичные	$O(n^2)$	10^{12}	11,6 дня
Кубические	$O(n^3)$	10^{18}	32000 лет
Экспоненциальные	$O(2^n)$	10^{301030}	В 10^{301006} раз больше, чем время существования вселенной

При условии, что единицей времени для нашего компьютера является микросекунда, компьютер может выполнить постоянный алгоритм за микросекунду, линейный — за секунду, а квадратичный — за 11,6 дня. Выполнение кубического алгоритма потребует 32 тысячи лет, что в принципе реализуемо.

Выполнение экспоненциального алгоритма практически невозможно, независимо от экстраполяции роста мощи компьютеров.

Временная сложность вскрытия алгоритма шифрования грубой силой пропорциональна количеству возможных ключей, которое экспоненциально зависит от длины ключа. Если n — длина ключа, то сложность вскрытия грубой силой равна

$O(2^n)$, и для шифротекста на основе DES-алгоритма при 56-битовом ключе составляет 2^{56}, а при 112-битовом ключе — 2^{112}. В первом случае вскрытие возможно, а во втором — нет [3].

Сложность проблем

Теория сложности также классифицирует и сложность самих проблем, а не только сложность конкретных алгоритмов решения проблемы. Проблемы можно разбить на классы в соответствии со сложностью их решения. Самые важные классы и их предполагаемые соотношения показаны на рис. 4.1.

Находящийся в самом низу класс **P** состоит из всех проблем, которые можно решить за полиномиальное время. Класс **NP** — из всех проблем, которые можно решить за полиномиальное время только на недетерминированной машине Тьюринга (машина Тьюринга — это теоретический компьютер, представляющий собой конечный автомат с бесконечной лентой памяти для чтения-записи). Недетерминированная машина Тьюринга способна делать предположения. Машина предполагает решение проблемы — либо «удачно угадывая», либо перебирая все предположения параллельно — и проверяет свое предположение за полиномиальное время.

Важность **NP** в криптографии состоит в следующем: многие симметричные алгоритмы и алгоритмы с открытыми ключами могут быть взломаны за недетерминированное полиномиальное время. Для данного шифротекста C криптоаналитик просто угадывает открытый текст X и ключ k и за полиномиальное время выполняет алгоритм шифрования со входами X и k и проверяет, равен ли результат C. Это имеет важное теоретическое значение, потому что устанавливает верхнюю границу сложности криптоанализа этих алгоритмов.

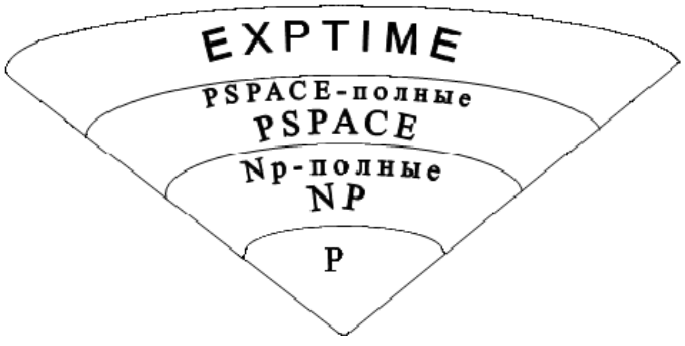


Рис. 4.1 – Классы сложности проблем

Класс **NP** включает класс **P**, так как любая проблема, решаемая за полиномиальное время на детерминированной машине Тьюринга, будет также решена за полиномиальное время на недетерминированной машине Тьюринга, при этом просто пропускается этап предположения.

Вопрос, верно ли **P**=**NP**, является центральным нерешенным вопросом теории вычислительной сложности. И не ожидается, что он будет решен в ближайшее время. Если кто-то покажет, что **P**=**NP**, то многие классы шифров тривиально взламываются за детерминированное полиномиальное время (детерминированными алгоритмами).

Следующим в иерархии сложности идет класс **PSPACE**. Проблемы класса **PSPACE** могут быть решены в полиномиальном пространстве, но не обязательно за полиномиальное время. **PSPACE** включает **NP**, но ряд проблем **PSPACE** кажутся сложнее, чем **NP**.

Существует класс проблем так называемых **PSPACE**-полных, обладающих следующим свойством: если любая из них является **NP**-проблемой, то **PSPACE**=**NP**, и если любая из них является **P**-проблемой, то **PSPACE**=**P**.

И наконец, существует класс проблем **EXPTIME**. Эти проблемы решаются за экспоненциальное время. Может быть действительно доказано, что **EXPTIME**-полные проблемы не могут быть решены за детерминированное полиномиальное время. Также показано, что **P** не равно **EXPTIME**.

NP-полные проблемы. Майкл Кэри (Michael Carey) и Дэвид Джонсон (David Johnson) составили список более чем 300 **NP**-полных проблем [6]. Среди них *проблема путешествующего коммивояжера*. Путешествующему коммивояжеру нужно посетить различные города, используя только один бак с горючим (существует максимальное расстояние, которое он может проехать). Существует ли маршрут, позволяющий ему посетить каждый город только один раз, используя этот единственный бак с горючим?

4.3 Теория чисел

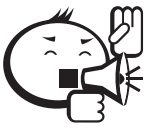
Арифметика вычетов

Иногда ее называют «арифметикой часов». Это арифметика по модулю 12. Двадцать три по модулю 12 равно 11 [6].

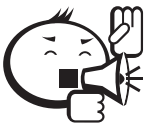
$$(10 + 13) \bmod 12 = 23 \bmod 12 = 11 \bmod 12.$$

Другим способом записать это является утверждение об эквивалентности 23 и 11 по модулю 12:

$$10 + 13 \equiv 11 \pmod{12}.$$



В основном $a \equiv b \pmod{n}$, если $a = b + kn$ для некоторого целого k [3, 5] . Если a неотрицательно и b находится между 0 и n, можно рассматривать b как остаток при делении a на n. Иногда b называется *вычетом a* по модулю n. Иногда a называется *конгруэнтным b* по модулю n (знак тройного равенства, ≡, обозначает конгруэнтность). Одно и то же можно сказать разными способами. Конгруэнтность в переводе с латинского языка означает соразмерный, соответствующий, совпадающий.



Множество чисел от 0 до $n - 1$ образует то, что называется *полным множеством вычетов* по модулю n. Это означает, что для любого целого a его остаток по модулю n является некоторым числом от 0 до $n - 1$.

Операция a mod *n обозначает остаток от a, являющийся некоторым целым числом от 0 до n* – 1. *Эта операция называется приведением по модулю.*

Пример

Например, 5 mod 3 = 2.

Это определение mod может отличаться от принятого в некоторых языках программирования.

Арифметика вычетов коммутативна, ассоциативна и дистрибутивна. Кроме того, приведение каждого промежуточного результата по модулю *n*дает тот же результат, как и выполнение всего вычисления с последующим приведением конечного результата по модулю n [6, 8].

$$(a + b) \bmod n = \big((a \bmod n) + (b \bmod n)\big) \bmod n,$$
$$(a - b) \bmod n = \big((a \bmod n) - (b \bmod n)\big) \bmod n,$$
$$(a \cdot b) \bmod n = \big((a \bmod n) \cdot (b \bmod n)\big) \bmod n,$$
$$\big(a \cdot (b + c)\big) \bmod n = \Big(\big((a \cdot b) \bmod n\big) + \big((a \cdot c) \bmod n\big)\Big) \bmod n.$$

Вычисление mod n часто используется в криптографии, так как вычисление дискретных логарифмов и квадратных корней mod n может быть нелегкой проблемой. Арифметика вычетов легче реализуется на компьютерах, поскольку она ограничивает диапазон промежуточных значений и результата и, кроме того, поз-

воляет выполнять возведение в степень без огромных промежуточных результатов. Вычисление степени некоторого числа по модулю другого числа,

$$a^x \bmod n,$$

представляет собой просто последовательность умножений и делений, но существуют приемы, ускоряющие это действие.

Так как операции дистрибутивны, быстрее выполнить возведение в степень как поток последовательных умножений, каждый раз получая вычеты. Например, если вы хотите вычислить $a^8 \bmod n$, не выполняйте семь умножений и одно приведение по модулю:

$$(a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a) \bmod n.$$

Вместо этого выполните три меньших умножения и три меньших приведения по модулю:

$$\left(\left(a^2 \bmod n\right)^2 \bmod n\right)^2 \bmod n.$$

Точно так же

$$a^{16} \bmod n = \left(\left(\left(a^2 \bmod n\right)^2 \bmod n\right)^2 \bmod n\right)^2 \bmod n.$$

Вычисление a^x, где x не является степенью 2, ненамного труднее, поскольку двоичная запись представляет x в виде степеней 2, и всегда можно сделать достаточно простые преобразования, например:

$$a^{25} \bmod n = \left(a \cdot a^{24}\right) \bmod n = \left(a \cdot a^8 \cdot a^{16}\right) \bmod n =$$

$$= \left(a \cdot \left(\left(a^2\right)^2\right)^2 \cdot \left(\left(\left(a^2\right)^2\right)^2\right)^2\right) \bmod n = \left(a\left(\left(\left(a \cdot a^2\right)^2\right)^2\right)^2\right) \bmod n.$$

Этот метод уменьшает количество операций, в среднем до $1,5 \cdot k$ операций, где k — длина числа x в битах.

Операция, обратная возведению в степень по модулю n, вычисляет *дискретный логарифм* [6].

Простые числа

Простым называется целое число, большее единицы, единственными множителями которого являются 1 и оно само: оно не делится ни на одно другое число. Два — это простое число. Простыми являются и 73, 2521, 2365347734339 и $2^{756839} - 1$ [3, 5]. Существует бесконечно много простых чисел. Криптография, особенно криптография с открытыми ключами, часто использует большие простые числа (512 бит и даже больше).

Наибольший общий делитель

Два числа называются взаимно простыми, если у них нет общих множителей, кроме 1. Иными словами, если наибольший общий делитель a и n равен 1 [6]. Это записывается в виде

$$\text{НОД}(a, n) = 1.$$

Взаимно просты числа 15 и 28, 13 и 500. А числа 15 и 27 не являются взаимно простыми. Простое число взаимно просто со всеми другими числами, кроме чисел, кратных данному простому числу.

Одним из способов вычислить наибольший общий делитель двух чисел является *алгоритм Евклида*. Евклид описал этот алгоритм в своей книге «Элементы», написанной в 300 году до нашей эры. Историки считают, что этот алгоритм лет на 200 старше. Это самый древний нетривиальный алгоритм, который дошел до наших дней, и он все еще применяется [6, 9].

Обратные значения по модулю

Обратное значение для $4 - 1/4$, потому что $1 = 4 \cdot 1/4$, в общем случае $1 = a \cdot 1/a$ [6]. В мире вычетов проблема усложняется:

$$4 \cdot x = 1 \pmod 7.$$

Это уравнение эквивалентно обнаружению x и k, таких, что $4x = 7k + 1$, где x и k — целые числа. Общая задача состоит в нахождении x, такого, что

$$1 = (a \cdot x) \bmod n,$$

где x и n — целые числа. Это также можно записать как $a^{-1} \equiv x \pmod n$.

Проблему обратных значений по модулю решить нелегко. Иногда у нее есть решение, иногда нет. Например, обратное значение 5 по модулю 14 равно 3. С другой стороны, у числа 2 нет обратного значения по модулю 14.

В общем случае у уравнения $a^{-1} \equiv x \pmod n$ существует единственное решение, если a и n взаимно просты. Если a и n не являются взаимно простыми, то уравнение не имеет решений. Если n является простым числом, то любое число от 1 до $n - 1$ взаимно просто с n и имеет в точности одно обратное значение по модулю n.

Существует два способа определения обратного значения a по модулю n. Один из них — вычисление обратного значения a по модулю n с помощью расширенного алгоритма Евклида приведен у Кнута.

Малая теорема Ферма. Если m — простое число и a не кратно m, то *малая теорема Ферма* утверждает [6]

$$a^{m-1} \equiv 1 \pmod m.$$

(Пьер де Ферма (Pierre de Fermat), французский математик, жил с 1601 по 1665 год.)

Функция Эйлера. Существует другой способ вычислить обратное значение по модулю n, но его не всегда возможно использовать.

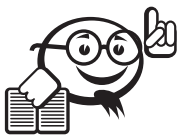
Приведенным множеством остатков $\bmod n$ *называется подмножество полного множества остатков, члены которого взаимно просты с n.*

Например, приведенное множество остатков $\bmod 12$ — это $\{1, 5, 7, 11\}$. Если n — простое число, то приведенное множество остатков $\bmod n$ — это множество всех чисел от 1 до $n - 1$. Для любого n, не равного 1, число 0 никогда не входит в приведенное множество остатков.

Функция Эйлера, которую также называют функцией фи Эйлера и записывают как $\phi(n)$*, — это количество элементов в приведенном множестве остатков по модулю n. Иными словами,* $\phi(n)$ *— это количество положительных целых чисел, меньших n и взаимно простых с n (для любого n, большего 1) [6, 8].*

(Леонард Эйлер (Leonhard Euler), швейцарский математик, жил с 1707 по 1783 год).

Если n — простое число, то $\phi(n) = n - 1$. Если $n = pq$, где p и q — простые числа, то $\phi(n) = (p - 1)(q - 1)$. Эти числа появляются в некоторых алгоритмах с открытыми ключами.



Теорема Эйлера. При $n > 1$ и НОД$(a, n) = 1$ имеем $a^{\phi(n)} \equiv 1 \pmod n$. Теперь легко вычислить, что для простого n, и при условии, что a не делится на n, справедлива формула [6]

$$a^{n-1} \equiv 1 \pmod n.$$

Разложение на множители

Разложить число на множители — значит, найти его простые сомножители. Например:

$$10 = 2 \cdot 5$$

$$60 = 2 \cdot 2 \cdot 3 \cdot 5$$

$$252601 = 41 \cdot 61 \cdot 101.$$

Разложение на множители является одной из древнейших проблем теории чисел [6, 8].

Единственность разложения на простые сомножители:

1) Всякое целое число a или взаимно просто с данным простым числом p, или делится на p. Действительно, здесь существует только два указанных варианта.

2) Если произведение нескольких сомножителей делится на p, то по крайней мере один из сомножителей делится на p.

3) Всякое целое число, большее 1, разлагается на произведение простых сомножителей, и притом единственным способом, если отвлечься от порядка следования сомножителей: $a = p_1 \cdot p_2 \cdot \ldots \cdot p_n$.

4) В разложении числа a на простые сомножители некоторые из них могут повторяться. Обозначая буквами $p_1, p_2, \ldots, p_k$ различные из них и буквами $\alpha_1, \alpha_2, \ldots, \alpha_k$ кратность их вхождения в a, получим каноническое разложение числа a на сомножители: $a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ [8].

Пример

Пример: каноническое разложение числа $588000 = 2^5 \cdot 3 \cdot 5^3 \cdot 7^2$.

Сегодня самым лучшим алгоритмом разложения на множители является *решето числового поля чисел* (Number field sieve, *NFS*) [6].

Решето общего числового поля — это самый быстрый из известных алгоритмов для чисел размером 110 и более разрядов. В своем первоначальном виде он был непрактичен, но за последние несколько лет последовательно улучшен. Перечислим некоторые другие алгоритмы, вытесненные NFS.

Квадратичное решето (Quadratic sieve, *QS*). Это самый быстрый из известных и чаще всего используемых алгоритмов для чисел, длина которых меньше 110 десятичных разрядов. Более быстрая версия этого алгоритма называется *множественным полиномиальным квадратичным решетом*.

Алгоритм Монте-Карло Полларда (Pollard's Monte Carlo algorithm). Этот алгоритм приведен у Кнута [9].

Проверка делением (Trial division). Этот самый старый алгоритм разложения на множители состоит из проверки каждого простого числа, меньшего или равного квадратному корню из раскладываемого числа.

4.4 Генерация простого числа

Основное определение простого числа было дано в предыдущем пункте. Прежде чем перейти непосредственно к рассмотрению любого алгоритма генерации простого числа, следует остановиться на некоторых свойствах простого числа и дать несколько дополнительных определений [8].

1) Число 1 имеет только один положительный делитель, а именно единицу. В этом отношении число 1 в ряде натуральных чисел стоит особо. Всякое

целое число, большее 1, имеет не менее двух делителей — 1 и самого себя; если этими делителями исчерпываются все положительные делители целого числа, то оно — простое. Целое число, большее 1, имеющее, кроме 1 и самого себя, другие положительные делители, называется составным.

2) Наименьший, отличный от 1 делитель целого числа, большего 1, есть число простое.

3) Наименьший, отличный от 1 делитель составного числа a (согласно п. 2 он будет простым) не превосходит $\sqrt{a}$. Действительно, пусть q — этот делитель, тогда $a = q \cdot a_1$, $a_1 \geqslant q$, откуда, перемножая и сокращая на a_1, получим $a \geqslant q^2$, $q \leqslant \sqrt{a}$.

4) Число простых чисел бесконечно велико.

5) Для составления таблицы простых чисел, не превосходящих данного N, существует простой способ, называемый решетом Эритосфена. Выписываем числа:

$$1, 2, \ldots, N. \tag{4.1}$$

Первое число этого ряда, большее 1, есть 2, оно делится только на единицу и на самое себя, следовательно, оно простое. Из ряда (4.1) вычеркнем (как составные) все числа, кратные 2. Первое, следующее за 2, невычеркнутое число будет 3; оно не делится на 2 (иначе оно оказалось бы вычеркнутым). Следовательно, 3 делится только на 1 и на самого себя, а поэтому оно также будет простым. Теперь вычеркнем из ряда (4.1) все числа, кратные 3, кроме самого 3. Первое, следующее за 3, невычеркнутое число будет 5, оно не делится ни на 2, ни на 3 (иначе оно оказалось бы вычеркнутым). Следовательно, 5 делится только на 1 и самого себя, а потому оно также будет простым. И так далее. Когда указанным способом уже вычеркнуты все числа, кратные простых, меньших простого p, то все невычеркнутые числа, меньшие p^2, будут простые. Действительно, всякое составное a, меньшее p^2, нами уже вычеркнуто как кратное его наименьшего простого делителя, который $\leqslant \sqrt{a}$, $< p$. Отсюда следует:

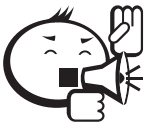
а) приступая к вычеркиванию кратных простого числа p, вычеркивание необходимо начинать с числа p^2;

б) составление таблицы простых чисел $\leqslant N$ закончено, как только вычеркнуты все составные кратные простых, не превосходящих $\sqrt{N}$.

Для алгоритмов с открытыми ключами нужны простые числа. Их нужно множество для любой достаточно большой сети. Для чисел, близких к n, вероятность того, что случайно выбранное число окажется простым, равна $1/(\ln n)$. Поэтому полное число простых чисел, меньших чем n, равно $n/(\ln n)$ [6].

Существует ряд способов генерации простых чисел [6] *Solovay-Strassen* — алгоритм вероятностной проверки простоты числа, разработанный Робертом Соловэй (Robert Solovay) и Фолькером Штрассеном (Volker Strassen); *Rabin-Miller* — еще один, повсеместно используемый простой алгоритм, разработанный Майклом Рабином (Michael Rabin).

4.5 Дискретные логарифмы в конечном поле



В качестве однонаправленной функции в криптографии часто используется возведение в степень по модулю. Легко вычислить $a^x \bmod n$.

Задачей, обратной возведению в степень по модулю, является поиск дискретного логарифма. А это уже нелегкая задача — найти x, для которого $a^x \equiv b \pmod{n}$.

Например: если $3^x \equiv 15 \bmod 17$, то $x = 6$.

Решения существуют не для всех дискретных логарифмов. Следует отметить, что речь идет только о целочисленных решениях. Легко заметить, что следующее уравнение не имеет решений $3^x \equiv 7 \pmod{13}$. Еще сложнее решать эту задачу для 1024-битовых чисел [6].

Безопасность многих алгоритмов с открытыми ключами основана на задаче поиска дискретных логарифмов, поэтому эта задача была глубоко изучена. Если p является простым числом и используется в качестве модуля, то сложность поиска дискретных логарифмов в мультипликативной группе полей простых чисел по существу соответствует разложению на множители числа n того же размера, где n — это произведение двух простых чисел приблизительно равной длины [6].

Стивен Полиг (Stephen Pohlig) и Мартин Хеллман нашли способ быстрого вычисления дискретных логарифмов при условии, что $p - 1$ раскладывается на малые простые множители. По этой причине в криптографии используются только такие поля, для которых $p - 1$ обладает хотя бы одним большим простым множителем. Вычисление дискретных логарифмов тесно связано с разложением на множители. Если вы можете решить проблему дискретного логарифма, то вы можете и разложить на множители. Истинность обратного утверждения никогда не была доказана. В настоящее время существует три метода вычисления дискретных логарифмов в поле простого числа [6]: линейное решето, схема целых чисел Гаусса и решето числового поля.

Контрольные вопросы по главе 4

1) Охарактеризуйте понятие энтропии сообщения.
2) Каким образом норма языка выражается через энтропию и длину сообщения?
3) Определите понятие абсолютной нормы языка.
4) Охарактеризуйте термин «избыточность языка».
5) Чему равна энтропия криптосистемы?
6) Определите понятие расстояния уникальности.

7) Приведите выражение для расстояния уникальности в случае симметричных криптоалгоритмов.
8) Опишите основные методы маскировки избыточности открытого текста сообщения.
9) Какими параметрами описывается вычислительная сложность алгоритма?
10) Охарактеризуйте полиномиальные и экспоненциальные алгоритмы.
11) Опишите классы сложности проблем.
12) Охарактеризуйте операцию приведения по модулю n.
13) Опишите способ получения таблицы простых чисел на основе решета Эритосфена.

Глава 5

КОМПЬЮТЕРНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ

5.1 Симметричные шифры

Работа симметричных шифров включает в себя два преобразования:

$$C = E_k(m) \text{ и } m = D_k(C),$$

где m — открытый текст, E — шифрующая функция, D — расшифровывающая функция, k — секретный ключ, C — шифротекст. Следует отметить, что как шифрующая, так и расшифровывающая функции общеизвестны, и тайна сообщения при известном шифротексте зависит только от секретности ключа k.

Для обеспечения хорошей криптостойкости симметричных алгоритмов шифрования необходимо предусмотреть достаточно большое пространство ключей. Принято считать, что вычисления, состоящие из 2^{80} шагов, в ближайшие несколько лет будут неосуществимы. Поэтому ключ, исключающий взлом простым перебором, должен насчитывать, по крайней мере, 80 битов [5].

Упрощенная модель шифрования

На рис. 5.1 изображена упрощенная модель шифрования битовой строки, которая, несмотря на свою простоту, вполне подходит для практического применения [5]. Идея модели состоит в применении к открытому тексту обратимой операции для получения шифротекста, а именно побитовое сложение по модулю 2 открытого текста со «случайным потоком» битов. Получатель может восстановить текст с помощью обратной операции, сложив шифротекст с тем же самым случайным потоком.

Такую модель легко реализовать на практике, поскольку для ее реализации необходима одна из простейших компьютерных операций — исключающее ИЛИ, т. е. сложение по модулю 2, которое обозначается знаком «⊕». Шифруя каждое новое сообщение своим ключом, длина которого совпадает с длиной открытого текста, мы получим абсолютно стойкую симметричную криптосистему называемую одноразовым блокнотом. Однако, несмотря на совершенство этого алгоритма, он не применяется на практике, поскольку порождает почти неразрешимую проблему распределения ключей.

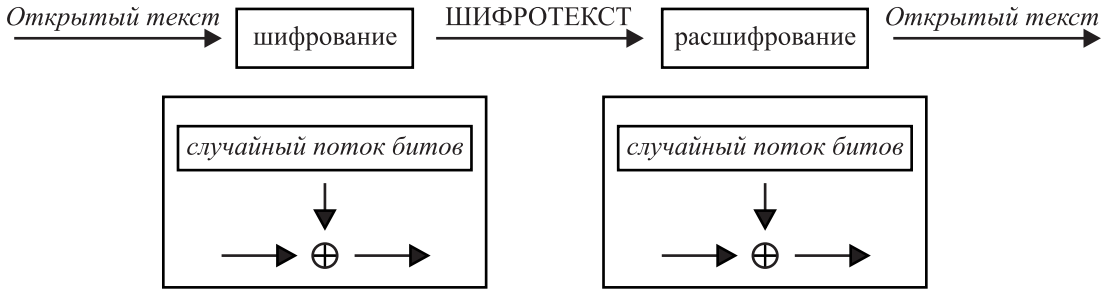


Рис. 5.1 – Упрощенная модель, шифрующая строку битов

В связи с этим разрабатываются симметричные криптосистемы, в которых длинное сообщение шифруется коротким ключом, причем этот ключ можно использовать несколько раз. Естественно, такие системы далеки от абсолютно стойких, но, с другой стороны, распределение ключей для них — хотя и трудная, но вполне решаемая задача.

Как уже отмечалось, большинство симметричных шифров можно разделить на две больших группы. Первая — поточные шифры, где за один раз обрабатывается один элемент данных (бит или буква), а вторая — блочные шифры, в которых за один шаг обрабатывается группа элементов данных (например, 64 бита).

5.2 Поточные шифры

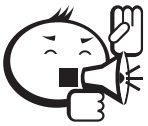


Рисунок 5.2 дает простую иллюстрацию поточного шифра. Обратите внимание, как она похожа на предыдущую упрощенную модель. Тем не менее случайный поток битов теперь генерируется по короткому секретному ключу с помощью открытого алгоритма, называемого *генератором ключевого потока.* Здесь биты шифротекста получаются по правилу: $C_i = m_i \oplus k_i$, где $m_0, m_1, \ldots$ — биты открытого текста, а $k_0, k_1, \ldots$ — биты ключевого потока [5].

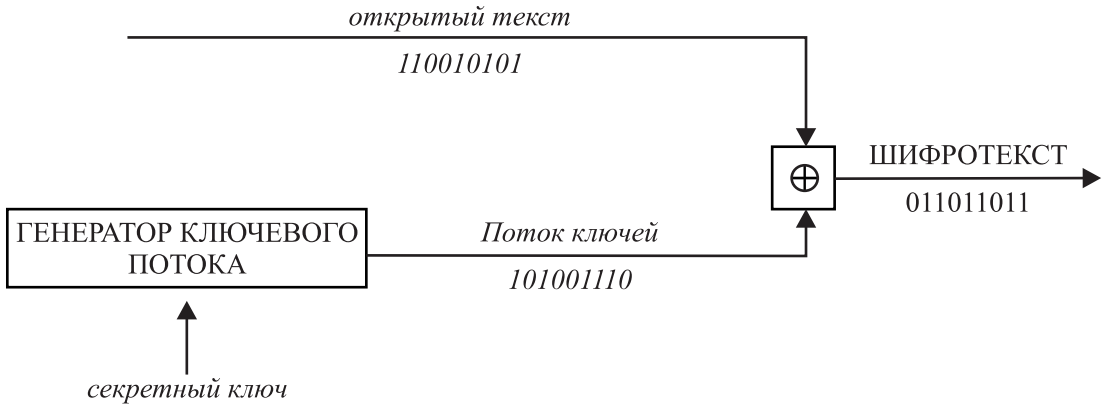


Рис. 5.2 – Поточные шифры

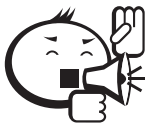
Поскольку процесс шифрования — это сложение по модулю 2, расшифрование является, по существу, той же самой операцией:

$$m_i = C_i \oplus k_i.$$

Поточные шифры, похожие на описанные выше, просты и удобны для реализации. Они позволяют очень быстро шифровать большой объем данных. Поэтому они подходят для передачи аудио- и видео-сигналов в реальном времени. Кроме того, в этом процессе не происходит накопления ошибки. Если отдельный бит шифротекста исказился в процессе передачи вследствие слабого радиосигнала или из-за вмешательства противника, то в расшифрованном открытом тексте только один бит окажется неверным. Однако повторное использование того же ключа дает тот же ключевой поток, что влечет за собой зависимость между соответствующими сообщениями. Предположим, например, что сообщения m_1 и m_2 были зашифрованы одним ключом k. Тогда противник, перехватив шифровки, легко найдет сумму по модулю 2 открытых текстов:

$$C_1 \oplus C_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2.$$

Следовательно, необходимо менять ключи либо с каждым новым сообщением, либо с очередным сеансом связи.



Чтобы придать необходимую стойкость шифру, генератор ключевого потока производит строку битов с определенными свойствами.

Как минимум, ключевой поток должен:

- Иметь большой период. Поскольку ключевой поток получается в результате детерминированного процесса из основного ключа, найдется такое число n, что $k_i = k_i + n$ для всех значений i. Число n называется периодом последовательности и для обеспечения стойкости шифра выбирается достаточно большим.
- Иметь псевдослучайные свойства. Генератор должен производить последовательность, которая кажется случайной. Другими словами, генерируемая последовательность должна выдержать определенное число статистических тестов на случайность.
- Обладать линейной сложностью. Далее объясняется значение этого термина.

5.3 Блочные шифры

На рис. 5.3 изображена схема блочного алгоритма шифрования.

Блочный шифр за один прием обрабатывает блок открытого текста. Основное отличие блочного шифра от поточного состоит в том, что поточным шифрам необходимо постоянно помнить о том, какое место битовой строки они в данный момент обрабатывают, чтобы определить, какую часть ключевого потока нужно сейчас генерировать; блочные же шифры избавлены от этой необходимости [5].

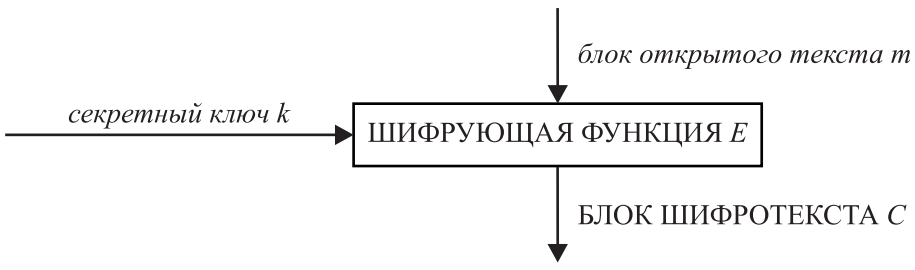
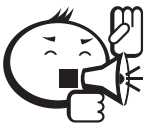


Рис. 5.3 – Схема работы блочного шифра

Размер блока для шифрования обычно выбирают разумно большим. В системе *DES* (стандарт шифрования данных), например, он состоит из 64 битов, а в современных блочных криптосистемах он достигает 128 битов и более.



Часто зашифрованный первый блок сообщения используют для шифрования следующего. Такой прием обычно называют *режимом шифрования.* Режимы используются, чтобы избежать некоторых атак, основанных на стирании или вставке, придавая каждому блоку шифротекста контекст, присущий всему сообщению.

Сегодня принято на вооружение довольно много разновидностей блочных шифров, некоторые из которых с большой долей вероятности используются Вашим web-браузером: *RC5*, *RC6*, *DES* или *3DES*. Наиболее знаменитый из них — *DES*, т. е. стандарт шифрования данных.

Впервые он был опубликован в середине семидесятых годов XX века как федеральный стандарт США и вскоре оказался, де-факто, международным стандартом в банковских операциях. *DES* успешно выдержал испытание временем, но к началу 90-х годов назрела необходимость в разработке новых стандартов. Произошло это потому, что как длина блока (64 бита), так и размер ключа (56 битов) оригинального алгоритма *DES* оказались недостаточными для обеспечения секретности сообщений. В настоящее время можно восстановить 56-битовый ключ системы *DES*, используя либо сеть компьютеров, либо специализированные аппаратные средства ЭВМ.

В ответ на эту проблему национальный институт стандартов и технологий США (NIST) положил начало своего рода соревнованию по поиску нового блочного шифра, достойного названия «новый стандарт шифрования» (*Advanced Encryption Standard, AES*) [5].

В отличие от фактически засекреченных работ над проектированием *DES* проект *AES* осуществлялся публично. Множество исследовательских групп всего мира представили свои варианты *AES* на конкурс. В финал вышли пять алгоритмов, которые изучались глубже с целью выбора победителя.

Это были криптосистемы:

- *MARS* от группы при компании *IBM*;
- *RC6*, представленная компанией *RSA Security*;
- *Twofish* от группы, базирующейся в Коунтерпэйне, Беркли и других местах;
- *Serpent* от группы трех ученых, работающих в Израиле, Норвегии и Британии;
- *Rijndael* от двух бельгийских криптографов.

В конце 2000 г. NIST объявил, что победителем конкурса был выбран шифр *Rijndael*.

Криптосистема *DES* и все финалисты проекта *AES* — примеры *итерированного* блочного шифра. В таких шифрах стойкость обеспечивается повторяющимся использованием простой *раундовой функции,* преобразующей n-битовые блоки в n-битовые блоки, где n — размер блока шифра. Число раундов S может меняться или быть фиксированным. Как правило, с увеличением числа раундов уровень стойкости блочного шифра повышается.

При каждом применении раундовой функции используется подключ

$$k_i \text{ при } 1 \leqslant i \leqslant S,$$

выводящийся из основного секретного ключа k с помощью алгоритма *разворачивания ключа.* Чтобы шифротекст можно было успешно расшифровать, функция, генерирующая подключи, должна быть обратимой. При расшифровании подключи применяются в порядке, обратном тому, в котором они использовались при шифровании. Требование обратимости каждого раунда не подразумевает обратимости функций, в нем участвующих. На первый взгляд это кажется странным, но станет совершенно очевидным после подробного обсуждения криптосистемы *DES.* Функции, которые в ней используются, необратимы, но тем не менее каждый раунд обратим. В то же время, в схеме *Rijndael* обратимы не только раунды, но и все функции.

Представляет интерес обсудить вопрос, какой из шифров лучше — блочный или поточный? По-видимому, корректного ответа не существует. Они оба используются и обладают разными свойствами:

- Блочный шифр является более общим, и его легко трансформировать в поточный.
- Поточный шифр имеет более математизированную структуру, что, с одной стороны, дает больше возможностей для его взлома, но с другой — позволяет легче изучать и строго оценивать его стойкость.
- Общие поточные шифры не очень удобны с точки зрения программного обеспечения, так как они обычно шифруют один бит за прием. Однако они высоко эффективны с точки зрения аппаратной реализации.
- Блочные шифры удобны как для программных, так и для аппаратных средств, но они не допускают такой высокой скорости обработки информации, как поточные.
- Аппаратные средства функционируют быстрее, чем программное обеспечение, но этот выигрыш происходит за счет снижения гибкости.

5.4 Шифр Фейстеля

Шифр *DES* — вариант базисного шифра Фейстеля (см. рис. 5.4), названного по имени Г. Фейстеля, работавшего в фирме *IBM* и выполнившего некоторые из самых ранних невоенных исследований в области алгоритмов шифрования [5]. Интересная особенность шифра Фейстеля заключается в том, что функция раунда обратима вне зависимости от свойств функции F (см. рис. 5.4).

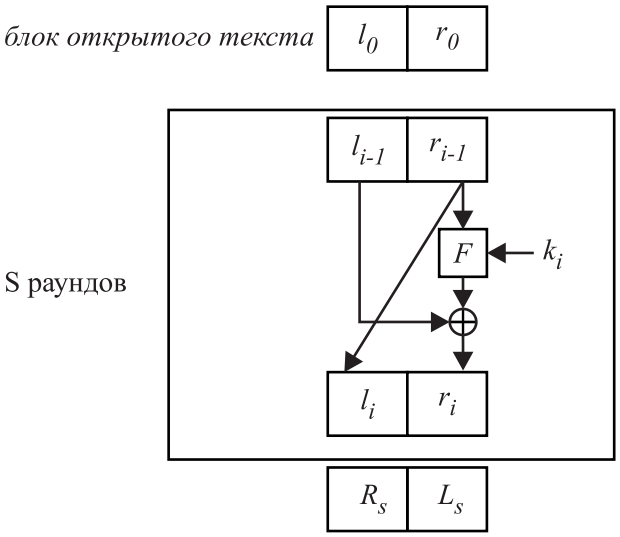


Рис. 5.4 – Основные операции шифра Фейстеля

Чтобы в этом убедиться, обратимся к формулам. Каждый раунд шифрования осуществляется по правилу

$$l_i = r_{i-1}, \quad r_i = l_{i-1} \oplus F(k_i, r_{i-1}).$$

Следовательно, расшифрование происходит за счет преобразований:

$$r_{i-1} = l, \quad l_{i-1} = r_i \oplus F(k_i, l_i).$$

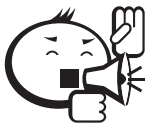
Характерные особенности предлагаемой схемы шифрования заключаются в следующем:

- в качестве F можно выбрать любую функцию и получить шифрующую функцию, которая будет обращаться при помощи секретного ключа;
- одну и ту же микросхему можно использовать как для шифрования, так и для расшифрования. Нам необходимо лишь проследить за порядками подключей, которые в этих процессах обратны друг другу.

Для создания криптостойкого шифра необходимо решить следующие вопросы:

- как генерировать подключи,
- сколько должно быть раундов,
- как определить функцию F.

5.5 Шифр DES



Работа над *DES* была начата в начале 1970-х годов группой сотрудников *IBM*, в которую входил и Фейстель [5]. Отправной точкой проекта послужил более ранний шифр — «Люцифер», как называли его в *IBM*.

Было известно, что управление национальной безопасности (*NSA*) внесло изменения в проект. Долгое время специалисты в области секретности считали, что изменения состояли в использовании ловушки в функции F. Однако теперь считается, что они были направлены на повышение безопасности шифра.

В документах национального института стандартизации США (*ANSI*) криптосистема *DES* называется *алгоритмом шифрования данных* (*DEA*), а международная организация по стандартизации, ссылаясь на шифр *DES*, пользуется аббревиатурой *DEA-1*. Этот алгоритм являлся мировым стандартом на протяжении более чем двадцати лет и утвердился как первый доступный всем желающим официальный алгоритм.

Основные черты шифра DES определяются прежде всего тем, что он — обобщение шифра Фейстеля и, кроме того, в нем:

- *число раундов S равно 16,*
- *длина блока n — 64 бита,*
- *размер ключа — 56 битов,*
- *каждый из подключей $k_1, k_2, \ldots, k_{16}$ насчитывает 48 битов.*

Заметим, что для многих современных алгоритмов длина ключа в 56 битов недостаточна. Поэтому в *DES* зачастую используют три ключа вместо одного, проводя три итерации стандартного процесса. При этом, как легко подсчитать, длина ключа становится равной 168 битам.

Такая версия классического шифра называется *тройным DES* или *3DES* (рис. 5.5).

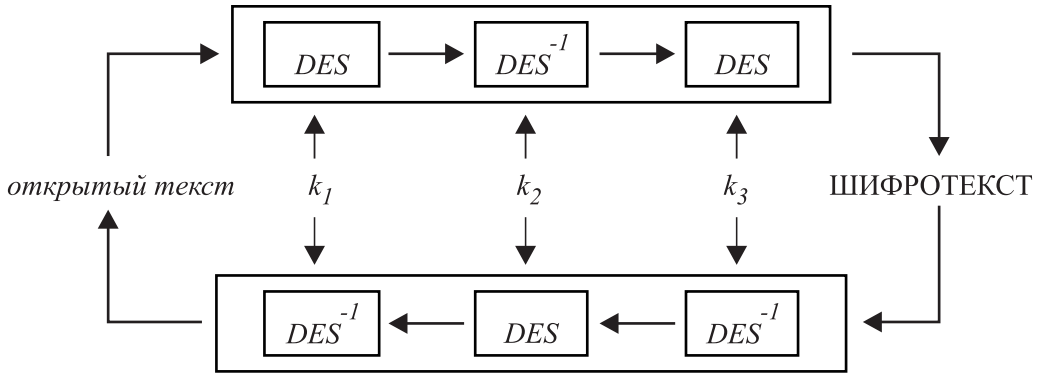


Рис. 5.5 – Тройной DES

Есть и другой способ модификации *DES*, в которой берут два ключа, увеличивая длину основного ключа до 112 битов.

Как уже отмечалось, в первом приближении *DES* — это шифр Фейстеля с 16 раундами (рис. 5.6), за исключением того, что как перед, так и после основных итераций алгоритма Фейстеля осуществляются некоторые перестановки. На рис. 5.6 показано, что два блока меняются местами перед последней перестановкой алгоритма. Эта замена не влияет на стойкость шифра, и пользователи часто задавались вопросом: зачем ее вообще делать? Один из членов творческого коллектива, разработавшего *DES*, утверждал, что она облегчает микросхемную реализацию процедуры шифрования.

Шифр *DES* преобразует открытый текст из 64 битов следующим образом:

- производит начальную перестановку (IP);
- расщепляет блок на левую и правую половины;
- осуществляет 16 раундов с одним и тем же набором операций;
- соединяет половины блока;
- производит конечную перестановку.

Конечная перестановка обратна начальной. Это позволяет использовать одно и то же программное обеспечение и «железо» для двух сторон процесса: шифрования и расшифрования. Разворачивание ключа дает 16 подключей по 48 битов каждый, выделяя их из 56-битного основного ключа [5].

5.6 Режимы работы DES

Блочный шифр, подобный *DES* или *Rijndael*, можно по-разному использовать для шифрования строк данных. Вскоре после *DES* в США был принят еще один федеральный стандарт, *рекомендующий* четыре способа эксплуатации алгоритма *DES* для шифрования данных. С тех пор эти режимы стали общепринятыми и применяются с любыми блочными шифрами [5].

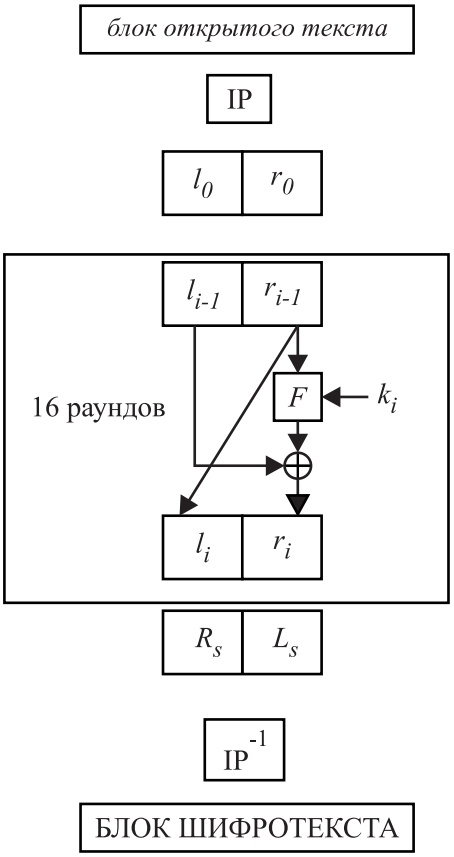


Рис. 5.6 – Алгоритм *DES*

Перечислим их:

- *ECB*. Этот режим прост в обращении, но слабо защищен от возможных атак с удалениями и вставками. Ошибка, допущенная в одном из битов шифротекста, влияет на целый блок в расшифрованном тексте.
- *CBC* — наилучший способ эксплуатации блочного шифра, поскольку предназначен для предотвращения потерь в результате атаки с использованием удалений и вставок. Здесь ошибочный бит шифротекста при расшифровании не только превращает в ошибочный блок, в котором содержится, но и портит один бит в следующем блоке открытого текста, что можно легко определить и интерпретировать как сигнал о предпринятой атаке.
- *OFB*. При таком методе блочный шифр превращается в поточный. Режим обладает тем свойством, что ошибка в один бит, просочившаяся в шифротекст, дает только один ошибочный бит в расшифрованном тексте.
- *CFB*. Как и в предыдущем случае, здесь блочный шифр трансформируется в поточный. Отдельная ошибка в криптограмме при этом влияет как на блок, в котором она была допущена, так и на следующий блок, как при режиме *CBC*.

Расссмотрим более подробно приведенные стандартные режимы шифрования [5].

Режим ECB

Режим ECB (Electronic Code Book — электронная кодовая книга) является простейшим среди стандартных способов использования блочного шифра. Данные m, которые предстоит зашифровать, делятся на блоки по n битов:

$$m_1, m_2, \ldots, m_q.$$

Последний из них, при необходимости, дополняют до длины n. По ним определяются блоки $C_1, \ldots, C_q$ как результат воздействия шифрующей функции

$$C_i = E_k(m_i),$$

как показано на рис. 5.7.

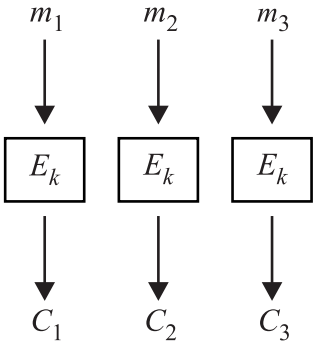


Рис. 5.7 – Шифрование в режиме *ECB*

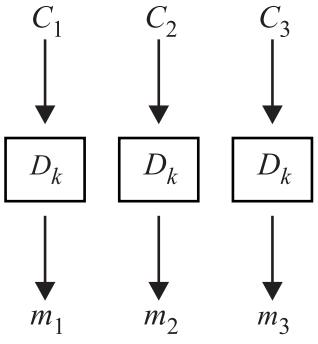


Рис. 5.8 – Расшифрование в режиме *ECB*

Расшифрование здесь — простое обращение предыдущей операции (см. рис. 5.8).

С режимом *ECB* связан ряд проблем. Первая возникает из-за того, что при равенстве $m_i = m_j$ мы получим одинаковые блоки шифротекста $C_i = C_j$, т. е. одинаковые блоки на входе индуцируют совпадающие блоки на выходе. Это, действительно, проблема, поскольку шаблонные начало и конец сообщений совпадают.

Вторая проблема связана с тем, что удаление из сообщения какого-либо блока не оставляет следов, и атакующий может таким образом исказить передаваемую информацию. Третья очень близка ко второй, но связана со вставкой блоков из других сообщений.

Таким атакам можно противостоять, добавляя контрольные суммы нескольких блоков открытого текста или используя режим, при котором к каждому блоку шифротекста добавляется «контекстный идентификатор».

Режим CBC

Один из путей обхода проблем, возникающих при использовании режима ECB, состоит в «зацеплении» шифра, т. е. в добавлении к каждому блоку шифротекста контекстного идентификатора. Самый простой способ сделать это — применить режим «сцепления блоков шифра» или CBC (сокращение от «Cipher Block Chaining»).

В этом режиме открытый текст, как обычно, разбивается на серию блоков:

$$m_1, m_2, \ldots, m_q$$

Как и в предыдущем режиме, последний блок может потребовать дополнения, чтобы длина открытого текста стала кратной длине блока. Шифрование осуществляется согласно формулам (см. рис. 5.9).

$$C_1 = E_k\left(m_1 \oplus IV\right), \quad C_i = E_k\left(m_i \oplus C_{i-1}\right) \text{ при } i > 1.$$

Следует отметить, что в вычислении первого блока шифротекста участвует величина IV (начальное значение), которую следует отнести к заданию шифрующей функции. Величину IV привлекают к шифрованию с тем, чтобы шифрованные версии одинаковых частей открытого текста выглядели по-разному. Нет необходимости скрывать значение IV, и на практике ее передают в открытом виде как часть сообщения.

Естественно, величина IV участвует и в расшифровании. Этот процесс выглядит следующим образом (рис. 5.10):

$$m_1 = D_k\left(C_1\right) \oplus IV, \quad m_i = D_k\left(C_i\right) \oplus C_{i-1} \text{ при } i > 1.$$

Напомним, что при шифровании в режиме *ECB* ошибка в одном знаке шифротекста, появляющаяся на стадии передачи сообщения, повлияет на весь блок, в котором она допущена, и он, естественно, будет расшифрован неверно. В случае режима *CBC*, как видно из формул, ошибочный знак повлияет не только на свой блок, но и на соответствующий бит в следующем блоке.

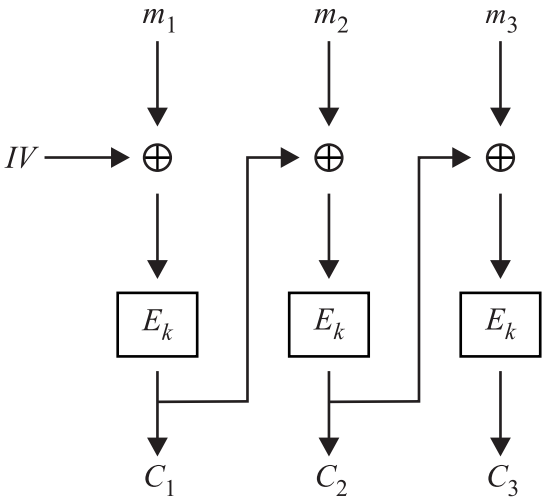


Рис. 5.9 – Шифрование в режиме *CBC*

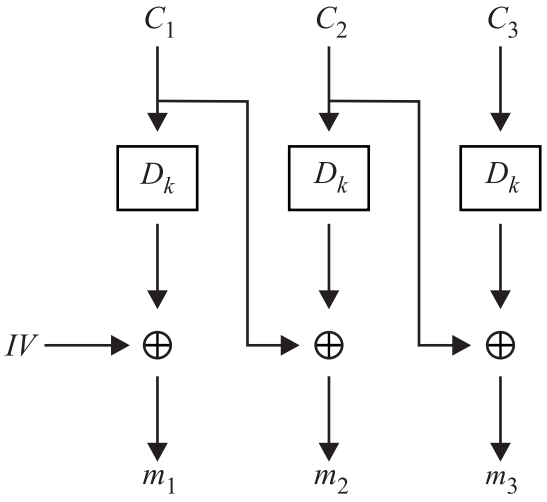


Рис. 5.10 – Расшифрование в режиме *CBC*

Режим OFB

Режим, называемый «обратной связью по выходу», или OFB (Output Feedback), адаптирует блочный шифр к его поточному использованию. Для этого выбирается переменная j ($1 \leqslant j \leqslant n$*), обозначающая число битов на выходе генератора потока ключей при каждой итерации. С помощью блочного шифра создается поток ключей, j битов за один раз. Рекомендуется брать j, равное n, поскольку при этом ожидаемая длина периода потока ключей получается большей, нежели при других значениях.*

Как и ранее, мы разбиваем открытый текст на серию блоков:

$$m_1, m_2, \ldots, m_q.$$

Но на этот раз, в отличие от предыдущих случаев, блоки состоят из j битов. Процесс шифрования происходит по следующей схеме (рис. 5.11). Прежде всего, переменной x_1 присваивается начальное значение IV. Затем, при $i = 1, 2, \ldots, q$, делаются преобразования: $y_i = E_k(x_i)$, $e_i = j$ крайних слева битов блока y_i, $C_i = m_i \oplus e_i$, $x_{i+1} = y_i$.

Расшифрование происходит аналогично (см. рис. 5.12).

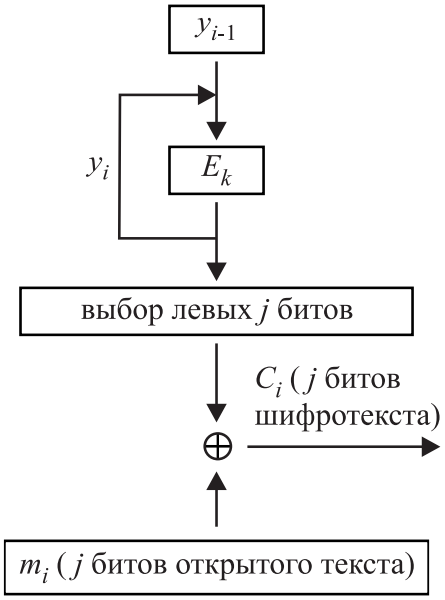


Рис. 5.11 – Шифрование в режиме *OFB*

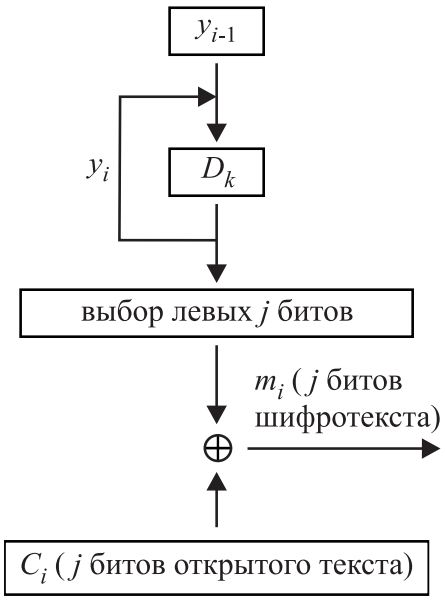


Рис. 5.12 – Расшифрование в режиме *OFB*

Режим CFB

Последний режим, который мы рассмотрим, носит название «обратной связи по шифротексту», или CFB (Cipher FeedBack). Он похож на режим OFB, но блочный шифр в нем трансформируется в поточный.

Напомним, что в предыдущем режиме начало потока ключей получается из значения IV, а остальной поток формируется пошагово, в результате шифрования значения шифрующей функции, вычисленного на предыдущей стадии. В случае *CFB* поток ключей возникает в результате еще одного шифрования блоков криптограммы (рис. 5.13): $y_0 = IV$, $z_i = E_k(y_{i-1})$, $e_i = j$ крайних слева битов блока z_i, $y_i = m_i \oplus e_i$.

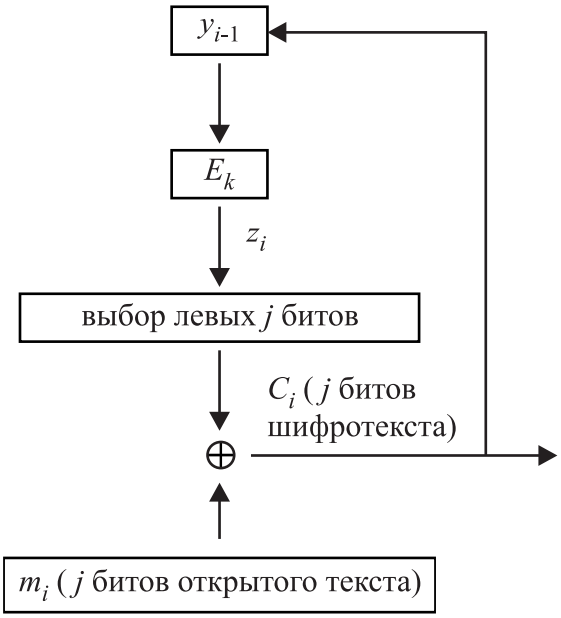
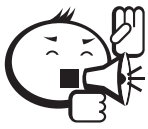


Рис. 5.13 – Шифрование в режиме *CFB*

5.7 Шифр Rijndael

Победитель конкурса *AES*, объявленный в конце 2000 года, алгоритм *Rijndael*, был разработан двумя бельгийскими криптографами: Дименом (Daemen) и Рийменом (Rijmen). Эта криптосистема, относясь к блочным шифрам, имеет много общего с *DES*, хотя и не является непосредственным обобщением шифра Фейстеля. Для обеспечения криптостойкости алгоритм *Rijndael* включает в себя повторяющиеся раунды, каждый из которых состоит из замен, перестановок и прибавления ключа [5].



Rijndael — настраиваемый блочный алгоритм, который может работать с блоками из 128, 192 или 256 битов. Применяются ключи шифрования трех фиксированных размеров 128, 192 и 256 битов В зависимости от размера ключа конкретный вариант алгоритма обозначается как AES-128, AES-192, AES-256. Для каждой комбинации блока и размера ключа определено свое количество раундов. Наиболее часто используемый вариант алгоритма, при котором блоки, как и ключ, состоят из 128 битов. В этом случае в алгоритме выполняется 10 раундов. При этом осуществляется вычисление 10 подключей $K_1, \ldots, K_{10}$, каждый из которых включает в себя четыре 32-битовых слова.

Существует множество компьютерных алгоритмов шифрования. Ниже рассматриваются наиболее часто применяемые алгоритмы [3–6].

5.8 Алгоритм криптографического преобразования ГОСТ 28147-89

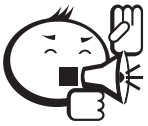
Настоящий стандарт (ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИИ ГОСТ 28147-89) устанавливает единый алгоритм криптографического преобразования для систем обработки информации в сетях электронных вычислительных машин (ЭВМ), отдельных вычислительных комплексах и ЭВМ, который определяет правила шифрования данных и выработки имитовставки.

Алгоритм криптографического преобразования предназначен для аппаратной или программной реализации, удовлетворяет криптографическим требованиям и по своим возможностям не накладывает ограничений на степень секретности защищаемой информации [3].

Стандарт обязателен для организаций, предприятий и учреждений, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях ЭВМ, в отдельных вычислительных комплексах или в ЭВМ.

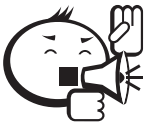
ГОСТ 28147-89 — отечественный стандарт на шифрование данных. Стандарт включает три алгоритма шифрования (дешифрирования) данных: режим простой замены, режим гаммирования, режим гаммирования с обратной связью и режим выработки имитовставки. С помощью имитовставки можно зафиксировать случайную или умышленную модификацию зашифрованной информации. Вырабатывать имитовставку можно или перед шифрованием (после дешифрирования) всего сообщения, или одновременно с шифрованием (дешифрированием) по блокам. При этом блок информации шифруется первыми шестнадцатью циклами в режиме простой замены, затем складывается по модулю 2 со вторым блоком, результат суммирования вновь шифруется первыми шестнадцатью циклами и т. д.

Алгоритмы шифрования ГОСТ 28147-89 обладают достоинствами других алгоритмов для симметричных систем и превосходят их своими возможностями. Так, ГОСТ 28147-89 (256-битовый ключ, 32 цикла шифрования) по сравнению с такими алгоритмами, как DES (56-битовый ключ, 16 циклов шифрования) и FEAL-1 (64-битовый ключ, 4 цикла шифрования), обладает более высокой криптостойкостью за счет более длинного ключа и большего числа циклов шифрования.



Следует отметить, что, в отличие от DES, у ГОСТ 28147-89 блок подстановки можно произвольно изменять, то есть он является дополнительным 512-битовым ключом, что позволяет увеличить криптостойкость алгоритма.

5.9 Стандарт симметричного шифрования данных IDEA



IDEA (International Data Encryption Algorithm, международный алгоритм шифрования данных) является блочным шифром, он работает с 64-битовыми блоками открытого текста. Длина ключа — 128 битов. Для шифрования и дешифрирования используется один и тот же алгоритм.

Как и другие блочные шифры, IDEA использует и запутывание, и рассеяние. Философия, лежащая в основе проекта, представляет собой «объединение операций из различных алгебраических групп» [6]. Смешиваются три алгебраических группы, и все они могут быть легко реализованы как аппаратно, так и программно:

- XOR;
- сложение по модулю 2^{16};
- умножение по модулю $2^{16} + 1$.

Все эти операции (а в алгоритме используются только они, перестановки на битовом уровне не применяются) работают с 16-битовыми подблоками. Этот алгоритм даже эффективнее на 16-битовых процессорах. IDEA является частью PGP [3, 6].

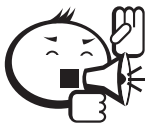
5.10 Однонаправленная хэш-функция MD5

MD5 (Message Digest, краткое изложение сообщения) — это однонаправленная хэш-функция.

MD5 — это улучшенная версия MD4. Хотя она сложнее MD4, их схемы похожи, и результатом MD5 также является 128-битовое хэш-значение. После некото-

рой первоначальной обработки MD5 обрабатывает входной текст 512-битовыми блоками, разбитыми на шестнадцать 32-битовых подблоков. Выходом алгоритма является набор из четырех 32-битовых блоков, которые объединяются в единое 128-битовое хэш-значение. Во-первых, сообщение дополняется так, чтобы его длина была на 64 бита короче числа, кратного 512. Этим дополнением является 1, за которой вплоть до конца сообщения следует столько нулей, сколько нужно. Затем к результату добавляется 64-битовое представление длины сообщения (истинной, до дополнения). Эти два действия служат для того, чтобы длина сообщения была кратна 512 битам (что требуется для оставшейся части алгоритма) и чтобы гарантировать, что разные сообщения не будут выглядеть одинаково после дополнения [3, 6].

5.11 Асимметричный алгоритм шифрования данных RSA



RSA — первый полноценный алгоритм с открытым ключом, который можно использовать для шифрования и цифровых подписей [4–6].

Криптосистема RSA разработана в 1977 году и получила название в честь ее создателей: Рона Райвеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Эйдельмана (Leonard Adleman). Они воспользовались тем фактом, что нахождение больших простых чисел в вычислительном отношении осуществляется легко, но разложение на множители произведения двух таких чисел практически невыполнимо.

Доказано (теорема Рабина), что раскрытие шифра RSA эквивалентно такому разложению. Поэтому для любой длины ключа можно дать нижнюю оценку числа операций для раскрытия шифра, а с учетом производительности современных компьютеров — оценить и необходимое на это время. Возможность гарантированно оценить защищенность алгоритма RSA стала одной из причин популярности этой криптосистемы на фоне десятков других схем.

Поэтому алгоритм RSA используется в банковских компьютерных сетях, особенно для работы с удаленными клиентами (обслуживание кредитных карточек). В настоящее время алгоритм RSA используется во многих стандартах, среди которых SSL, S-HHTP, S-MIME, S/WAN, STT и PCT.

Основными математическими результатами, положенными в основу этого алгоритма, являются: малая теорема Ферма и функция Эйлера (раздел 4.3).

Рассмотрим небольшой пример, иллюстрирующий применение алгоритма RSA.

Пример

Зашифруем сообщение «CAB».

Для простоты будем использовать маленькие простые числа (на практике применяются гораздо большие). Выберем $p = 3$ и $q = 11$. Определим $n = 3 \cdot 11 = 33$. Найдем $(p-1)(q-1) = 20$.

Следовательно, в качестве d, взаимно просто с 20, например, $d = 3$. Выберем число e. В качестве такого числа может быть взято любое число, для которого удовлетворяется соотношение $(e \cdot 3) \pmod{20} = 1$, например 7.

Представим шифруемое сообщение как последовательность целых чисел с помощью отображения: $A \to 1$, $B \to 2$, $C \to 3$. Тогда сообщение принимает вид $3, 1, 2$.

Зашифруем сообщение с помощью ключа $\{7, 33\}$:

$$\left(3^7\right) \pmod{33} = 2187 \pmod{33} = 9,$$
$$\left(1^7\right) \pmod{33} = 1 \pmod{33} = 1,$$
$$\left(2^7\right) \pmod{33} = 128 \pmod{33} = 29.$$

Расшифруем полученное зашифрованное сообщение $(9, 1, 29)$ на основе закрытого ключа $\{3, 33\}$:

$$\left(9^3\right) \pmod{33} = 729 \pmod{33} = 3,$$
$$\left(1^3\right) \pmod{33} = 1 \pmod{33} = 1,$$
$$\left(29^3\right) \pmod{33} = 24389 \pmod{33} = 2.$$

Итак, в реальных системах алгоритм RSA реализуется следующим образом: каждый пользователь выбирает два больших простых числа (p, q) и в соответствии с описанным выше алгоритмом выбирает два простых числа e и d. Как результат умножения первых двух чисел (p, q) устанавливается n, $\{e, n\}$ образует открытый ключ, а $\{d, n\}$ — закрытый (хотя можно и наоборот).

Открытый ключ публикуется и доступен каждому, кто желает послать владельцу ключа сообщение, которое зашифровывается указанным алгоритмом. После шифрования сообщение невозможно раскрыть с помощью открытого ключа. Владелец же закрытого ключа без труда может расшифровать принятое сообщение.

В настоящее время алгоритм RSA активно реализуется как в виде самостоятельных криптографических продуктов, так и в качестве встроенных средств в популярных приложениях.

Важная проблема практической реализации — генерация больших простых чисел.

В конце 1995 года удалось практически реализовать раскрытие шифра RSA для 500-значного ключа. Для этого с помощью сети Интернет было задействовано 1600 компьютеров. Сами авторы RSA рекомендуют использовать следующие размеры

модуля n: 768 бит — для частных лиц; 1024 бит — для коммерческой информации; 2048 бит — для особо секретной информации.

Третий немаловажный аспект реализации RSA — вычислительный. Ведь приходится использовать аппарат длинной арифметики. Если используется ключ длиной k бит, то для операций по открытому ключу требуется $O(k^2)$ операций, по закрытому ключу — $O(k^3)$ операций, а для генерации новых ключей требуется $O(k^4)$ операций. По сравнению с алгоритмом DES, RSA требует в тысячи и десятки тысяч раз большее время.

5.12 Комплекс криптографических алгоритмов PGP

Общие сведения

Комплекс криптоалгоритмов PRETTY GOOD PRIVACY (PGP, весьма хорошая секретность) — программа для ведения секретной переписки, ставшая стандартом де-факто в гражданской криптографии. PGP представляет собой комплект программ, позволяющий шифровать и подписывать электронные сообщения [3, 6].

Реализации PGP имеются для большинства операционных систем:

- Windows (95,NT,2000, XP);
- UNIX (Linux, FreeBSD, и др.);
- MAC;
- MS-DOS;
- BeOS;
- VAX/VMS и других.

PGP умеет встраиваться в популярные почтовые программы для Windows: Outlook, OutlookExpress, Eudora, The Bat, что делает его использование достаточно легким для пользователя.

PGP — это свободно распространяемая программа безопасной электронной почты. Разработана Филипом Циммерманном (Philip Zimmermann), а выпущена фирмой Phil's Pretty Good Software [6].

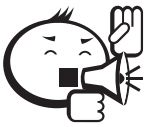
PGP является криптографической системой с высокой степенью секретности. PGP позволяет пользователям обмениваться файлами или сообщениями:

- с использованием функций секретности;
- с установлением подлинности;
- с высокой степенью удобства.

Шифрование сообщений и электронная подпись реализованы в PGP на основе технологии шифрования с открытым ключом. Для пользователя это значит, что у него (и у каждого, кто пользуется системой) имеется 2 ключа: открытый (или публикуемый) — public key, закрытый (или частный) — private key.

Публикуемый ключ пользователь раздает тем, с кем ведет шифрованную переписку. Этот ключ можно публиковать где угодно — он не содержит пароля.

Тот, кто будет писать шифрованное письмо, должен иметь публикуемый ключ своего адресата. Сам ключ представляет собой текстовый файл не очень понятного содержания.



Частный ключ — это ключ, который пользователь хранит у себя и никому не показывает. Только имея свой частный ключ, пользователь расшифровывает электронные письма и убеждается в подлинности подписи.

Конечно, PGP — это не единственная система шифрования корреспонденции (известна также, например, система VeriSign). Однако технология PGP имеет несколько неоспоримых *преимуществ.*

- Технология имеется для большинства операционных систем (многоплатформенность).
- Комплект программ является бесплатным и свободно распространяется для всех операционных систем.
- Технология не привязана к какому-либо центральному серверу. Открытые ключи можно передавать как угодно.
- Использовать PGP очень просто, поскольку он встраивается в почтовые программы.

В связи с тем что алгоритм шифрования с общим ключом значительно медленнее, чем стандартное шифрование с одним ключом, шифрование сообщения лучше выполнять с использованием быстрого высококачественного стандартного алгоритма шифрования с одним ключом.

В процессе, невидимом для пользователя, временный произвольный ключ, созданный только для этого одного «сеанса», используется для традиционного шифрования файла открытого текста. Тогда общий ключ получателя используется только для шифровки этого временного произвольного стандартного ключа. Зашифрованный ключ «сеанса» посылается, наряду с зашифрованным текстом (назы-

ваемым «ciphertext» — зашифрованный), получателю. Получатель использует свой собственный секретный ключ, чтобы восстановить этот временный ключ сеанса, и затем применяет его для выполнения быстрого стандартного алгоритма декодирования с одним ключом, чтобы декодировать все зашифрованное сообщение [3, 6].

Алгоритмы секретных ключей

PGP предлагает на выбор различные алгоритмы секретных ключей. Под *алгоритмом секретного ключа* понимается симметричный блочный шифр, использующий один ключ для шифрования и дешифрирования.

Могут быть использованы следующие три симметричных блочных шифра:

- CAST,
- Triple-DES,
- IDEA.

Все три шифра работают с 64-битовыми блоками открытого текста и шифротекста. Длина ключа для CAST и IDEA — 128 битов, в то время как Triple-DES использует 168-битовый ключ. Подобно Data Encryption Standard (DES), любой из этих шифров может быть использован в режимах CFB и CBC.

Шифровальный алгоритм CAST (CAST был разработан в Канаде Карлайслом Адамсом (Carlisle Adams) и Стаффордом Таваресом (Stafford Tavares)) был включен в PGP благодаря длине ключа (128-бит), быстроте работы, свободному распространению.

Следует отметить, что начиная с версии PGP 7.0 в комплекс криптоалгоритмов добавлен алгоритм Брюса Шнайера Twofish, вышедший в финал конкурса на улучшенный стандарт шифрования AES.

Распределенный подход к управлению ключами

Самой интересной особенностью PGP является *распределенный подход к управлению ключами*. Центров сертификации ключей нет, вместо этого в PGP поддерживается «сеть доверия». Каждый пользователь сам создает и распространяет свой открытый ключ. Пользователи подписывают ключи друг друга, создавая взаимосвязанное сообщество пользователей PGP. Таким образом, распределенное управление ключами реализуется с помощью поручителей.

Варианты PGP

Одной из свободно распространяемых версией PGP является 6.0.2. 7-я версия PGP называется «PGP Desktop» [3, 6]. В этой версии, кроме основного предназначения PGP, добавлены следующие модули (программы):

- PGP Disk — система шифрования данных на жестких дисках и других носителях.
- PGP IGO — система шифрования данных, передаваемых с помощью ICQ.
- PGP Net — система шифрования трафика и пакетный фильтр.

Существуют также 8 и 9 версии PGP.

Проблему защиты данных на винчестере очень удобно решает PGP Disk. На жестком диске создается файл, который при штатной работе отображается в логический диск. Вся информация в файле шифрованная, т. е. получив этот файл и не зная пароля, его практически невозможно расшифровать. Превращение файла в диск может сделать либо владелец ключа, либо человек, который ввел пароль. Система PGP Disk разработана под Windows-операционные системы.

Краткое изложение основных характеристик PGP

PGP — это протокол, который определяет правила взаимодействия существующих алгоритмов. PGP интегрирует известные криптосистемы для более надежной защиты. Реализацию PGP можно считать удавшейся, так как он до сих пор не был взломан.

PGP обладает следующими характеристиками:

- основывается на алгоритме RSA (шифрование с общим ключом);
- использует для различных целей такие алгоритмы, как IDEA, DES, Triple-DES, CAST, MD5;
- применяется распределенное управление ключами (с помощью поручителей);
- ключи хранятся в виде сертификатов ключей (в двух каталогах);
- шифрует и дешифрирует документы (файлы);
- можно обеспечить не только секретность сообщений, но и их подлинность (подпись сообщений);
- некоторые версии не могут быть использованы вне США.

Контрольные вопросы по главе 5

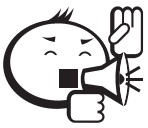
1) Опишите упрощенную модель шифрования битовой строки.
2) Приведите схему и опишите принцип работы поточного шифра.
3) Приведите схему и опишите принцип работы блочного шифра.
4) Перечислите алгоритмы — финалисты конкурса AES.
5) Охарактеризуйте основные операции шифра Фейстеля.
6) Охарактеризуйте основные операции шифра DES.
7) Опишите схему работы тройного DES.

8) Приведите основные характеристики шифра Rijndael.
9) Приведите режимы работы DES.
10) Охарактеризуйте режим применения блочного шифра ECB.
11) Охарактеризуйте режим применения блочного шифра CBC.
12) Охарактеризуйте режим применения блочного шифра OFB.
13) Охарактеризуйте режим применения блочного шифра CFB.
14) Охарактеризуйте стандарт шифрования ГОСТ 28147-89.
15) Опишите основные характеристики блочного шифра IDEA.
16) Приведите описание алгоритма с открытым ключом RSA.
17) Опишите основные свойства однонаправленной хэш-функции MD5.
18) Каково назначение комплекса криптоалгоритмов PGP?
19) Какие технологии шифрования применяются в PGP?

Глава 6

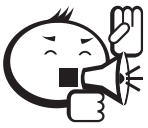
КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ И ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ КРИПТОГРАФИИ

6.1 Общие сведения



Особую актуальность использование криптографических средств защиты информации приобрело в связи с предоставлением ряда банковских услуг через Internet (Internet-banking), а также построением корпоративных виртуальных сетей передачи данных [2].

Активное развитие криптографических методов и инструментов способствовало появлению новой области человеческой деятельности — электронной коммерции.



Следует отметить, что при построении отечественных информационно-телекоммуникационных систем (ИТС) на основе импортных программно-аппаратных средств возникают проблемы с обеспечением безопасности таких систем [1, 2]. Объясняется это тем, что:

- импортные компоненты не могут считаться безопасными (доверенными);
- использование открытых сетей передачи данных позволяет проводить удаленные атаки.

Проблемы, связанные с обеспечением информационной безопасности в современных ИТС (имеющих как распределенный, так и локальный характер), обычно включают [2]:

1) Защиту информационных ресурсов локальной рабочей станции (ЛРС) от несанкционированного доступа.
2) Защиту в локальных сетях передачи данных.
3) Защиту межсетевого взаимодействия:
   - создание защищенных виртуальных сетей (VPN);
   - обеспечение защиты технологии клиент/сервер;
   - обеспечение защиты информационных ресурсов корпоративной сети, имеющей выход в общедоступные сети передачи данных, от атак извне;
   - обеспечение средств защиты пользовательского взаимодействия типа «абонент А — абонент В».
4) Защиту электронной почты и документооборота.
5) Защиту электронных платежных систем.

Встраивание программных средств защиты информации может осуществляться одним из следующих способов [2]:

- с использованием программных интерфейсов;
- с использованием криптосервера.

Основная проблема встраивания заключается в корректном использовании вызываемых функций.

Программным интерфейсом (далее API) называется детальное описание функций и используемых ими параметров. Формат обращения к функциям, описанный в программном интерфейсе, поддерживает прикладное ПО. Для того чтобы интегрировать защитный механизм в данное ПО, необходимо согласовать форматы функций (вызываемых и реализованных).

Наиболее известными API являются Crypto API, GSSAPI и SSPI. В ОС Windows NT 4.0 используется программный интерфейс Crypto API, дающий возможность шифрования (RC2, RC4, DES, RSA), выработки и проверки электронной надписи, однако в этой системе нет действующих программных средств с механизмами криптографической защиты [2]. Необходимо отметить, что алгоритмы шифрования, применяемые в Crypto API, используют укороченные ключи.

Подход к решению проблемы защиты на основе криптографического сервера заключается в локализации средства защиты информации, работающего с важнейшей информацией (секретные ключи, пароли и т. д.) на основе выделенной рабочей станции. В прикладное ПО, работающее на другой (находящейся на связи) рабочей станции, встраиваются только вызовы функций, реализованных в ПО криптосервера. Использование криптосервера позволяет выполняться средству защиты информации в доверенной программной среде.

Остановимся подробнее на существующей классификации уровней защиты информации.

Физический и канальный уровни

На этих уровнях в качестве механизмов безопасности обычно осуществляется шифрование соединения или трафика (зашифровываться может как весь трафик, так и выборочная часть), т. е. обеспечивается конфиденциальность передаваемой информации. В качестве вероятных угроз здесь можно выделить следующие [2]:

- несанкционированное подключение;
- ошибочная коммутация;
- прослушивание;
- перехват;
- фальсификация информации;
- имитоатаки;
- физическое уничтожение канала связи.

Для защиты информации на данном уровне обычно применяют шифрующие модемы, специализированные канальные адаптеры. К достоинствам реализации средств защиты на уровнях этого типа можно отнести:

- простоту применения;
- аппаратную реализацию;
- полную защиту трафика;
- прозрачность выполнения средствами защиты информации своих функций.

К недостаткам применения средств защиты на канальном или физическом уровне следует отнести [2]:

- негибкость решения (фиксированный тип канала, сложность адаптации к сетевой топологии, фиксированная производительность);
- низкую совместимость;
- высокую стоимость.

Сетевой уровень

При защите информации на сетевом уровне необходимо решить следующие задачи:

1) Защиту информации непосредственно в пакетах, передаваемых по сети.
2) Защиту трафика сети, то есть шифрование всей информации в канале.
3) Контроль доступа к ресурсам сети, т. е. защиту от нелегальных пользователей и от доступа легальных пользователей к информации, им не предназначенной.

Реализация средств защиты на сетевом уровне представляет больше возможностей для обеспечения безопасности передаваемых данных. Например, на данном уровне может быть реализована аутентификация рабочей станции, являющейся источником сообщений. Из угроз, специфичных для сетевого уровня, следует выделить:

- анализ служебной информации сетевого уровня, т. е. адресной информации и топологии сети;
- атаки на систему маршрутизации;
- фальсификации адресов; атаки на систему управления;
- прослушивание, перехват и фальсификацию информации;
- имитоатаки.

Для устранения перечисленных угроз применяются следующие решения [2]:

- пакетная фильтрация;
- административная защита на маршрутизаторах (в том числе шифрующих);
- протоколы защиты информации сетевого уровня (защита и аутентификация трафика);
- динамическое распределение сетевых адресов;
- защита топологии.

К достоинствам средств защиты на сетевом уровне относятся:

- полнота контроля трафика;
- универсальность;
- прозрачность;
- совместимость;
- адаптивность к сетевой технологии.

К недостаткам средств защиты на сетевом уровне можно отнести только неполноту контролируемых событий (неподконтрольность транспортных и прикладных протоколов). Сетевой уровень — это та ступень организации, на которой сеть становится полносвязной системой. На более низких уровнях защита может быть реализована только как набор двухточечных защищенных звеньев.

Только на сетевом уровне появляется возможность установления защищенного соединения между двумя компьютерами, расположенными в произвольных точках сети; на этой ступени появляется и понятие топологии, можно различить внешние и внутренние каналы. При сетевом взаимодействии реализуются такие возможности защиты информации, как фильтрация трафика между внутренней (корпоративной) сетью и внешней коммуникационной средой, защита от несанкционированного доступа из внешней сети во внутреннюю, маскировка топологий внутренних сетей.

Транспортный уровень

Самыми опасными угрозами на этом уровне следует считать [2]:

- несанкционированные соединения, разведку приложений;
- атаки на систему управления;
- прослушивание, перехват и фальсификацию информации;
- имитоатаки.

Традиционными решениями подобных проблем являются:

- защита в составе межсетевых экранов (контроль доступа к приложениям, управление доступом к серверам);
- proxy-системы;
- протоколы защиты транспортного уровня (защита и аутентификация данных).

Среди достоинств средств защиты на транспортном уровне можно выделить следующие:

- развитая функциональность;
- высокая гибкость системы.

Недостатками средств защиты являются [2]:

- неполнота защиты;
- неподконтрольность событий в рамках прикладных и сетевых протоколов.

Прикладной уровень

При создании и пересылке сложного электронного документа, состоящего из нескольких полей, необходимо обеспечить защиту какого-то отдельно взятого поля данных. Решение этой задачи может заключаться в применении средств криптографической защиты на прикладном уровне. Информация, зашифрованная на прикладном уровне, затем разбивается на пакеты и сетевые кадры, надежность доставки которых обеспечивается транспортной средой, — следовательно, криптографическая защита объекта прикладного уровня должна быть действительной и для всех нижестоящих уровней [2].

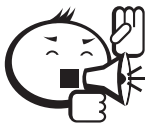
Необходимо отметить, что при защите информации на прикладном уровне процедуры передачи, разборки на пакеты, маршрутизации и обратной сборки не могут нанести ущерба конфиденциальности информации. Кроме того, на прикладном уровне существует возможность обеспечения аутентификации пользователя, породившего информацию.

Из основных угроз, возникающих на прикладном уровне, следует отметить:

- несанкционированный доступ к данным;
- разведку имен и паролей пользователей;
- атаки на систему разграничения прав доступа пользователей;
- маскировку под легитимного пользователя;
- атаки на систему управления, атаки через стандартные прикладные протоколы;
- фальсификацию информации;
- имитоатаки.

Традиционно применяемыми решениями по защите информации и информационных ресурсов прикладного уровня являются:

- встроенная защита приложений;
- межсетевые экраны с фильтрацией сетевых прикладных протоколов;
- proxy-системы и т. д. [2].



Криптографическая защита информации на прикладном уровне является наиболее предпочтительным вариантом защиты информации с точки зрения ее гибкости, однако эти меры могут оказаться наиболее сложными в части программно-технической реализации.

6.2 Обзор стандартов в области защиты информации

Международные стандарты

Особое место в современных ИТС занимают вопросы стандартизации методов и средств защиты информации и информационных ресурсов. Международные стандарты представлены серией ISO и ISO/IES и выпущены в свет Международной организацией по стандартизации (ISO) и Международной электротехнической комиссией (IES) [2].

ISO 7498-2 (X.800). Стандарт посвящен описанию архитектуры безопасности по отношению к модели взаимодействия открытых систем (OSI), включая описание расположения механизмов и служб безопасности на уровнях OSI.

ISO 8372. Этот стандарт описывает режимы блочного шифрования:

а) режим электронной книги (ECB);

б) режим сцепления блоков (CBC);

в) режим с обратной связью по шифротексту (CFB);

г) режим с обратной связью по выходу (OFB). Впервые был опубликован в 1987 г.

ISO 8730. Данный стандарт совместно со стандартом ISO 8731 описывает процедуру использования алгоритмов генерации MAC в банковских системах и является международным аналогом стандарта ANSI X9.9. В нем вводятся независимые от алгоритмов методы и требования использования MAC, включая форматы представления данных, а также способ, при помощи которого данный стандарт может быть применен к выбранному алгоритму.

ISO 8732. Стандарт посвящен управлению ключами в банковских системах и является аналогом стандарта ANSI X9.17.

ISO 9564. Стандарт посвящен методам управления и обеспечения безопасности персонального идентификационного номера (PIN). В части 9564-1 раскрываются принципы и средства защиты PIN в течение его жизненного цикла, позволяющие сохранить его втайне от злоумышленников, а часть 9564-2 посвящена использованию алгоритмов шифрования для защиты PIN.

ISO/IES 9594-8 (X.509). Данный стандарт посвящен двум типам аутентификации — простой и строгой. Представленная строгая аутентификация состоит из двух и трех передаваемых сообщений и основана на использовании ЭЦП и параметров, зависящих от времени. В стандарте также описывается процедура аутентичного распределения открытых ключей на основе сертификатов в формате X.509.

ISO/IES 9797. Этот стандарт описывает алгоритм генерации кодов аутентификации сообщений (MAC), основанный на использовании алгоритма блочного шифрования.

ISO/IES 9798. Данный стандарт состоит из пяти частей. В первой части (9798-1) описывается механизм аутентификации пользователей, основанный на использовании симметричного алгоритма шифрова-ния (9798-2), алгоритм генерации ЭЦП с асимметричными клю-чами (9798-3), криптографической функции проверки целостности или MAC (9798-4), некоторых дополнительных механизмов (9798-5).

ISO 11166. Стандарт состоит из нескольких частей и описывает асимметричные механизмы распределения симметричных ключей. Часть 11166-1 посвящена основным принципам и процедурам распределения ключей и форматам представления ключей, а также сертификации ключевой информации. Часть 11166-2 описывает применение RSA для шифрования и генерации ЭЦП.

ISO 11568. Стандарт описывает средства и процедуры управления ключами при финансовых операциях для симметричных и асимметричных алгоритмов.

ISO / IES 14888. Стандарт состоит из нескольких частей и посвящен построению схем ЭЦП. Часть 14888-1 знакомит с общепринятыми определениями и моделями построения схем ЭЦП. В части 14888-2 рассматриваются схемы ЭЦП, в которых ключ проверки подписи является результатом общедоступной функции от информации, идентифицирующей подписывающего. Часть 14888-3 посвящена распределению открытых ключей при помощи сертификатов открытых ключей.

Стандарты серии ANSI

В этом пункте перечислены стандарты, разработанные Американским национальным институтом стандартов (ANSI) и посвященные криптографическим алгоритмам и их применению в банковских системах.

ANSI X 9.92. Стандарт специфицирует DES-алгоритм, который в рамках этого стандарта представляется как алгоритм шифрования данных (Data Encryption Algorithm-DEA).

ANSI X 3.106. Стандарт описывает виды применения алгоритма DES.

ANSI X 9.9. Стандарт описывает применение MAC на основе использования DES в широком круге банковских систем.

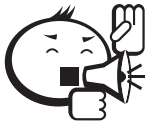
ANSI X 9.17. Данный стандарт основан на стандарте ISO 8732 и посвящен процедурам управления ключами, а также средствам распределения и защиты ключей применительно к банковским системам. В нем нашли отражение специфические аспекты использования ключей, такие как: счетчики ключей, преобразование ключей и подтверждение подлинности ключей.

ANSI X 9.19. Стандарт посвящен вопросам использования алгоритма DES для генерации MAC в конкретных типах банковских систем.

ANSI X 9.23. Стандарт описывает форматы представления данных при использовании алгоритма DES в банковских системах и методы обработки получаемых из каналов связи данных.

ANSI X 9.24. Данный стандарт описывает методы управления ключами, аутентификацию (основанную на DES) и шифрование PIN, ключей и других данных, а также руководящие принципы защиты ключей на всех этапах жизненного цикла.

ANSI X 9.28. Этот стандарт является продолжением стандарта ANSI X9.17 и описывает процедуры распределения ключей между пользователями, которые не имеют между собой и третьей стороной предварительного распределения ключевой информации.



ANSI X 9.31. Стандарт описывает алгоритм генерации и проверки ЭЦП на основе использования алгоритма RSA [2].

Государственные стандарты США

FIPS 46. Стандарт описывает DES-алгоритм.

FIPS 74. Стандарт описывает руководящие принципы применения и использования DES.

FIPS 113. Стандарт описывает алгоритм генерации MAC на основе использования DES.

Существует также большое количество стандартов в области безопасности Internet, например RFC, разработанный IETF (Internet Engineering Steering Group), и стандарты PKCS (Public-key Cryptography Standards) [2].

6.3 Подсистема информационной безопасности

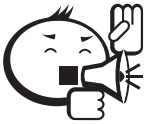
Понятие подсистемы информационной безопасности (ПИБ) включает в себя весь комплекс средств и мер по защите информации в ИТС.

Типовая ПИБ строится с учетом следующих принципов [2]:

1) Применяемые в ПИБ средства и технологии защиты должны обладать свойствами модульности, масштабируемости, открытости и возможности адаптации системы к различным условиям функционирования.
2) ПИБ ИТС должна предполагать полную независимость функционирования каждого из комплексов защиты. Нарушение функционирования любого компонента не должно приводить к изменению других характеристик защиты, тем более к отказу всей системы.
3) Применяемые средства и технологии должны обеспечивать открытость архитектуры, наращиваемость, а также предоставлять интерфейс для подключения внешних систем защиты информации.

Основные задачи подсистемы защиты локальных рабочих мест:

1) Разграничение доступа к ресурсам пользователей автоматизированных рабочих мест (АРМ) ИТС.
2) Криптографическая защита хранящейся на АРМ информации (конфиденциальность и целостность).
3) Аутентификация пользователей и авторизация их действий.
4) Обеспечение невозможности влияния программно-технического обеспечения АРМ на регламент функционирования прикладных процессов ИТС [2].



Основные задачи подсистемы защиты локальных вычислительных сетей (ЛВС). Для данной подсистемы характерна не только защита от несанкционированного доступа (НДС) к информации и информационным ресурсам в рамках ЛВС, но также и криптографическая защита информации, передаваемой в ЛВС на прикладном, системном уровне и на уровне структурированной кабельной системы.

Основные задачи подсистемы защиты межсетевого взаимодействия. Для этой подсистемы основным является защита ИТС от негативных воздействий со стороны внешних сетей передачи данных. Подсистема должна обеспечивать не только защиту информации (конфиденциальность и целостность), передаваемой во внешнюю среду, но также и защиту от внешнего разрушающего воздействия информации, находящейся в ИТС [2].

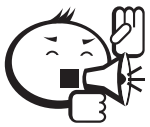
Основные задачи подсистемы аудита и мониторинга:

1) Осуществление централизованного, удаленного, автоматизированного контроля работы подконтрольных серверов, АРМ и других подсистем ПИБ.
2) Централизованное ведение протокола всех событий, происходящих на подконтрольных серверах, АРМах и подсистемах.
3) Выполнение автоматизированного анализа механизма принятия решений в нештатных ситуациях.

Основные задачи подсистемы технологической защиты:

1) Выработка принципов построения технологического процесса обработки информации в ИТС.
2) Определение контрольных точек технологического процесса.
3) Определение мест использования средств и методов защиты информации для регистрации, подтверждения и проверки прохождения информации через контрольные точки технологического процесса.
4) Исключение сосредоточения полномочий у отдельных должностных лиц [2].

6.4 Защита локальной рабочей станции



Под локальной рабочей станцией (ЛРС) подразумевается компьютер, не подсоединенный к каналам передачи данных. Любая защита в современных информационных системах начинается, прежде всего, с конкретного рабочего места. По статистике, большинство нарушений приходится именно на внутренних нарушителей [2].

Задачи обеспечения информационной безопасности осложняются тем, что функционирование средств защиты информации происходит в *недоверенной программной среде*. Имеется в виду среда, в которой невозможно однозначно доказать отсутствие влияния программного окружения на функционирование средств защиты информации. Недоверенность программной среды вытекает из того факта, что на сегодняшний день ПО, установленное на ЛРС, чаще всего зарубежного производства, функциональный состав которого в большинстве случаев является неопределенным. При этом под ПО подразумевается как прикладное программное обеспечение, так и обеспечение безопасности ОС.

Следует отметить, что наибольшую опасность представляют недокументированные возможности и закладки, находящиеся в ядре ОС. Так, например, в ОС Windows NT и в интерфейсе WIN 32 имеется ряд недокументированных возможностей [2].

Угрозы и задачи информационной безопасности для локальных рабочих станций

Потенциальных нарушителей можно разделить на следующие категории [2]:

- зарегистрированные пользователи ЛРС;
- имеющие доступ к штатным средствам управления ЛРС (клавиатура, устройства считывания информации и т. д.);
- пользователи, не имеющие физического доступа к ЛРС.

При этом нарушители обычно ставят перед собой следующие цели:

- компрометация секретной информации, в том числе и информации, относящейся к работе средств защиты информации ЛРС (ключевая информация, пароли пользователей и т. д.);
- изменение или уничтожение любой секретной информации;
- нарушение работоспособности всей системы в целом или ее отдельных компонентов.

Под нарушениями имеются в виду не только умышленные действия, но и неумышленные нарушения и ошибки обслуживающего персонала.

Обобщенный перечень угроз можно классифицировать по следующим признакам.

1) По характеру доступа:
   - с доступом к ПО;
   - с доступом только к аппаратным ресурсам;
   - без доступа к ЛРС.

Доступ к ПО подразумевает, что нарушитель имеет возможность взаимодействовать с установленным ПО, которое доступно для модификации и анализа. Доступ к аппаратному обеспечению подразумевает воздействие нарушителя на аппаратную часть с целью введения ее в режим, нарушающий надежное функционирование средств защиты информации. Отсутствие доступа к ЛРС подразумевает, что нарушитель имеет возможность воздействовать на систему только через каналы передачи информации, возникающие в ходе работы ЛРС. В качестве таких каналов можно отметить:

- электромагнитный канал;
- цепи заземления и питания;
- виброакустический канал.

Например, электромагнитный канал возникает вследствие появления электромагнитного излучения от ЛРС. Данное излучение модулируется в соответствии с процессами обработки информации, среди которых могут быть и процессы, обрабатывающие секретную информацию.

2) По характеру проявления:
   - активные
   - пассивные.

Активные проявления характеризуются тем, что нарушитель является инициатором негативных воздействий на систему (воздействия могут быть как умышленными, так и неумышленными); пассивные же угрозы заключаются в анализе информации, возникающей в ходе функционирования ЛРС.

3) По используемым в ходе реализации угрозы средствам:

- с использованием штатных средств, входящих в состав ЛРС;
- с использованием дополнительных угроз.

На рис. 6.1 представлена классификация угроз для однопользовательской ЛРС [2].

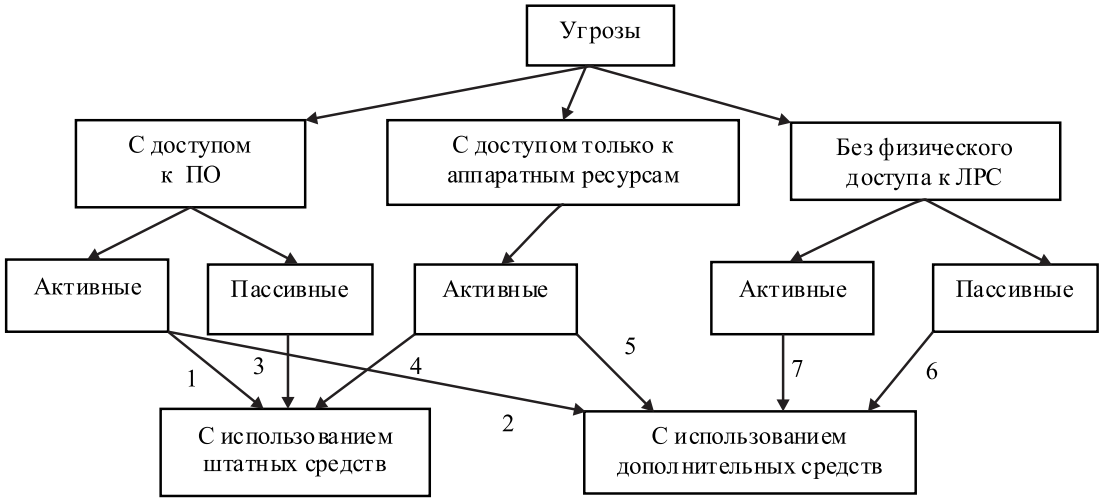


Рис. 6.1 – Классификация угроз

Ветвь классификации 1

Нарушитель, получивший доступ к ПО, может воспользоваться штатными средствами ПО для проведения атак. К штатным средствам относятся средства разработки и отладки ПО, входящие в состав некоторых типов прикладного ПО (например, Microsoft Word имеет в своем составе встроенный Visual Basic); средства управления и конфигурирования, входящие в состав программного обеспечения. Предполагается, что на ЛРС используются штатные средства защиты ОС, причем их конфигурация выполнена корректно. Угрозы, входящие в эту ветвь:

- создание программ-закладок, вирусов и программ, обеспечивающих прямой доступ к памяти или адресному пространству любого процесса, позволяющих обходить штатные средства защиты ОС и получать привилегии администратора ОС. С помощью подобных программ можно получать доступ к конфиденциальной информации за счет чтения файлов и областей памяти, в которых находятся секретные данные (пользовательский пароль, настройки системы), или нарушать целостность данной информации;
- нарушение целостности установленного прикладного или системного ПО с целью перевода штатных средств защиты в нештатный режим работы;

- удаление критической информации или части ПО, позволяющее нарушить работоспособность системы;
- внесение программ-закладок и вирусов в установленное ПО;
- оказание негативного влияния на работу аппаратных средств ЛРС;
- переконфигурирование системы с целью нарушения ее работоспособности;
- ошибочные действия персонала;
- анализ особенностей функционирования системы защиты с целью выявления некорректных состояний;
- доступ к остаточной информации, находящейся во временных файлах или в кластерах жесткого диска после удаления программ штатными средствами;
- удаление журналов аудита с целью скрыть факт воздействия на систему [2].

Ветвь классификации 2

В этом случае нарушитель использует свои программы проведения атак, и перечень угроз в данной ветви будет иметь тот же вид, что и ветвь классификации 1. Чтобы запустить подобные программы, нарушитель должен иметь возможность загрузить ПО с внешних носителей информации (гибкие диски, CD-ROM и т. д.). Также нарушитель может загрузить с внешних носителей свою версию ОС, и тогда последствия будут аналогичны тем, что описаны выше [2].

Ветвь классификации 3

Это пассивные угрозы, которые возможны только в случае отсутствия штатных механизмов защиты ОС или их некорректной настройки. К пассивным угрозам относятся:

- доступ к незащищенным файлам, содержащим критичную пользовательскую или системную информацию, с помощью штатных средств ПО;
- анализ функционирования ПО, установленного на ЛРС;
- анализ настроек ОС;
- добывание информации о содержащихся ошибках в ПО;
- анализ остаточной информации [2].

Ветвь классификации 4

Нарушитель имеет доступ к органам управления и средствам ввода/вывода информации в ЛРС, а также к контактным разъемам считывающих устройств, не будучи зарегистрированным штатным пользователем системы. Нарушитель может воздействовать через органы управления на работу ПО ЛРС, при этом считается, что он не имеет доступа внутрь корпуса ЛРС и получает возможность:

- загрузить свою версию ОС с внешних носителей информации;
- выдать себя за зарегистрированного пользователя системы с целью получить доступ к системному и/или прикладному ПО. После этого возникает

перечень угроз, перечисленных в первой, второй или третьей ветвях классификации;

- воздействовать на аппаратные средства с целью перевода их в некорректный режим функционирования, например заставить ПО работать на ЛРС, на которой не прошли тесты памяти в процессе загрузки;
- войти в режим конфигурирования BIOS;
- получить доступ к информации, относящейся к функционированию аппаратной части ЛРС;
- воздействовать на аппаратную часть с целью выведения из строя отдельных частей аппаратных компонентов ЛРС или всей ЛРС в целом [2].

Ветвь классификации 5

Данный перечень конкретных угроз состоит из тех же пунктов, что и предыдущий, с одним добавлением: получение физического доступа к аппаратным компонентам, осуществляющим хранение и обработку информации, нарушение целостности информации или изменение процессов обработки информации с целью доступа к конфиденциальной или критичной системной информации [2].

Ветвь классификации 6

Угрозы этого типа основаны на том, что аппаратные средства в ходе обработки информации создают побочные физические поля, законы изменения которых отражают процесс обработки и хранения информации. К таким полям относятся электромагнитное и виброакустическое. Также в ходе работы средств вычислительной техники могут возникнуть побочные каналы утечки информации, выражающиеся в наличии сопутствующих сигналов в линии электропитания или заземления.

По мере удаления от источника сигнал затухает, вследствие чего на определенном расстоянии побочный канал утечки информации не будет представлять опасности. Зона, в которой побочные сигналы могут представлять опасность, обычно контролируется, и доступ туда посторонним лицам (тем более нахождение аппаратуры, позволяющей фиксировать опасные сигналы) запрещен. Чтобы воспользоваться подобного рода возможностями, нарушитель должен обладать определенными средствами и навыками

- получения доступа к конфиденциальной пользовательской и системной информации;
- получения информации о процессах функционирования ПО и аппаратной части ЛРС [2].

Ветвь классификации 7

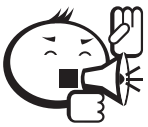
Угрозы этого типа отличаются от перечисленных выше следующим: нарушитель в ходе их реализации производит воздействие на ЛРС с помощью электромагнитных каналов и цепей питания и/или заземления. Факт наличия таких угроз

необходим при практической реализации дифференциального криптоанализа, заключающегося в вызове ошибочных ситуаций в процессах обработки и хранения информации, с целью последующего анализа полученного результата. Подобная атака может привести к компрометации секретных ключей пользователя. Таким образом, основная цель нарушителя в данном случае — вызвать сбой в аппаратной части ЛРС и вывести из строя ее отдельные компоненты.

При использовании ЛРС в многопользовательском режиме, т. е. при наличии нескольких зарегистрированных пользователей на ЛРС, возникают угрозы и со стороны легальных пользователей. Поскольку на одной ЛРС могут работать несколько пользователей, то на ней будет храниться и обрабатываться информация разных пользователей, в том числе и конфиденциальная. Рассматривать угрозы для данного случая будем с учетом предположения, что на ЛРС установлена ОС, имеющая штатные средства защиты информации и разграничения доступа к ресурсам. При отсутствии таких средств любой пользователь может:

- получить доступ к конфиденциальной информации других пользователей, хранящейся в незащищенных файлах;
- применить программы-закладки и вирусы, активизирующиеся в ходе работы на данной ЛРС другого пользователя;
- нарушить целостность пользовательской информации;
- получить доступ к системной информации, относящейся к работе других пользователей;
- выдать себя при регистрации в системе за другого пользователя и провести несанкционированные операции от его имени [2].

6.5 Методы и средства обеспечения информационной безопасности локальных рабочих станций



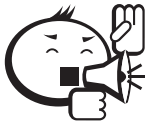
Создание защищенной системы является комплексной задачей. Конкретные средства защиты информации реализуют только типовые функции по ее защите. Реальная защитная система строится исходя из возможных угроз и выбранной политики безопасности.

На основе вышеизложенного перечня угроз можно предложить следующие меры по защите данных:

1) Обеспечение конфиденциальности и достоверности пользовательской информации. Реализуется на основе применения средств шифрования и ЭЦП, выполненных как в виде абонентского шифрования (отдельная программа-вызов, запуск которой предоставляет пользователю возможность применять функции шифрования в ЭЦП), так и в виде средств прозрачного шифрования. Например, вызовы функций шифрования встраиваются в используемое пользовательское ПО, в ходе работы которого незаметно для пользователя происходит вызов этих функций из прикладного ПО.

2) Разграничение доступа пользователей к информации, хранящейся и обрабатываемой на ЛРС при применении многопользовательского режима. Обычно используются штатные средства современных ОС.
3) Аутентификация пользователей и авторизация доступа к защищаемым объектам с использованием криптографических методов.
4) Контроль целостности секретной пользовательской или системной информации, основанный на применении программ генерации контрольных сумм и их проверки (бесключевые хэш-функции или имитовставки).
5) Контроль процесса загрузки ОС, при котором осуществляется загрузка строго определенной версии ОС и блокировка несанкционированной загрузки с внешних носителей или из сети.
6) Контроль целостности аппаратных ресурсов, который необходимо осуществлять непосредственно перед тем, как пользователь начнет работать с ЛРС.
7) Исключение негативного влияния на процесс функционирования пользовательского приложения со стороны программного окружения, установленного на ЛРС.
8) Контроль запуска задач пользователем, т. е. пользователю для запуска должны быть доступны только его задачи.
9) Защита от побочного электромагнитного излучения и утечки информации по цепям питания и заземления.
10) Физическое удаление остаточной информации, возникающей в ходе функционирования ПО (уничтожение временных файлов и т. д.).
11) Аудит и протоколирование работы средств защиты информации, системного и прикладного ПО.

Аудит



Аудит связан с действиями, затрагивающими безопасность системы.

К их числу относятся [2]:

- вход в систему (успешный или нет);
- выход из системы;
- обращение к удаленной системе;
- операции с файлами (открыть, закрыть, переименовать, удалить);
- смена привилегий или иных атрибутов безопасности (режима доступа и т. п.).

Полный перечень событий, подлежащих регистрации, зависит от выбранной политики безопасности и от общей спецификации системы.

При протоколировании события необходимо записывать:

1) дату и время события;
2) уникальный идентификатор;
3) отмечать пользователя, являющегося инициатором действия;
4) тип события;
5) результат действия (успех или неудача);
6) источник запроса (например, имя терминала);
7) имена затронутых объектов (например, файлов);
8) описания изменений, внесенных в базы данных защиты (например, новая метка безопасности объекта);
9) метки безопасности субъектов и объектов события [2].

При такой организации система аудита может фиксировать все события, связанные с функционированием прикладного и системного ПО. Анализ реализации подсистем аудита в современных ОС (операционных системах) показывает, что ядро системы (являющееся доверенным) взаимодействует с прикладным ПО (работа которого подлежит аудиту) через интерфейс обращений к данному сервису ОС и является наиболее удобным местом для осуществления мониторинга и аудита. Ядро ОС при обращении к нему прикладного ПО фиксирует события, подлежащие аудиту, а подсистема аудита заносит их в журнал событий, хранящийся на жестком диске.

Генерировать события может и прикладное ПО, но, в отличие от ядра ОС, данные действия не заносятся напрямую в журнал событий, а передаются в ядро системы, которое отправляет их в буфер (отведенный под запись событий) и впоследствии фиксирует в журнал событий. При осуществлении записи или чтения в данный буфер производится его блокировка с целью запрещения одновременного

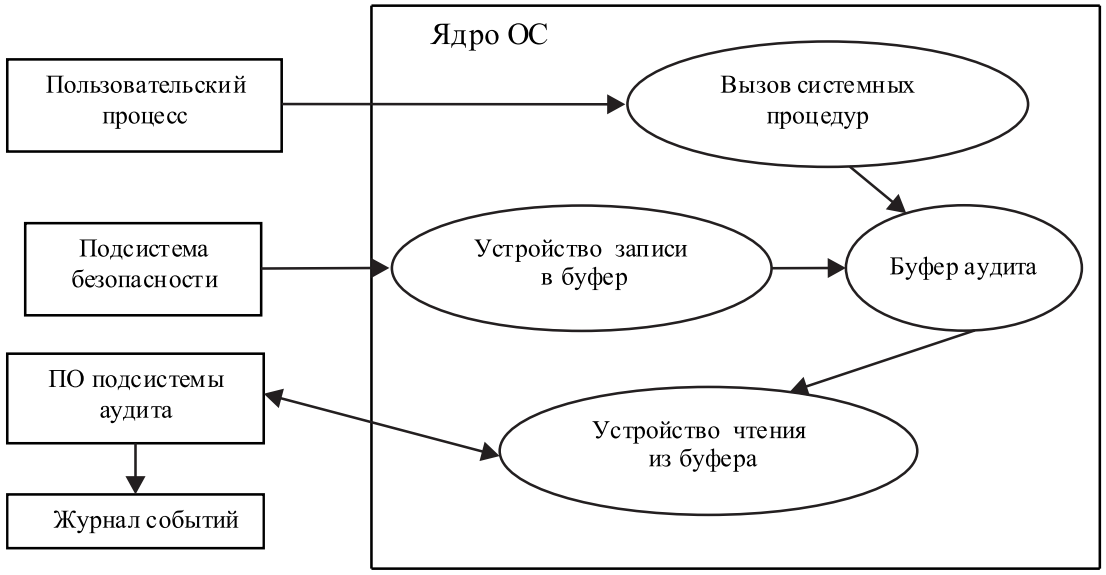


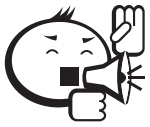
Рис. 6.2 – Архитектура системы аудита

чтения из него и записи в него. К каждому событию, помещенному в буфер, дописывается метка времени, последовательный номер и идентификатор процесса, породившего данное событие.

Общая схема архитектуры построения системы аудита представлена на рис. 6.2 [2].

Необходимо подчеркнуть важность не только сбора информации, но и ее регулярного и целенаправленного анализа. В этом плане удобно использовать средства аудита СУБД, поскольку к регистрационной информации могут естественным образом применяться SQL-запросы.

6.6 Защита в локальных сетях



Поскольку все глобальные сети передачи данных представляют собой объединение локальных сетей (ЛВС), имеющих различную топологию построения и использующих различные сетевые протоколы (TPC/IP, IPX/SPX, Netbios, DecNet и др.), безопасность ЛВС имеет первостепенное значение.

Общие вопросы безопасности в ЛВС

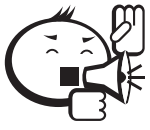
Среди основных угроз для ЛВС следует отметить следующие:

1) анализ сетевого трафика с целью получения доступа к конфиденциальной информации, например к передаваемым в открытом виде по сети пользовательским паролям;
2) нарушение целостности передаваемой информации;
3) получение несанкционированного доступа к информационным ресурсам;
4) попытка совершения ряда действий от имени зарегистрированного пользователя в системе.

Обеспечение защиты в соответствии с заданной политикой безопасности в ЛВС является комплексной задачей и осуществляется при помощи:

1) корректного администрирования сетевых настроек ОС;
2) дополнительных защитных механизмов — шифрования, ЭЦП, аутентификации сторон и т. д.;
3) организационных методов защиты и контроля за их неукоснительным соблюдением, например с использованием системы аудита.

Безопасность сетей Novell NetWare

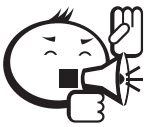


В ОС Novell NetWare возможны атаки типа «человек-в-середине» при условии, что нарушитель способен просматривать пакеты, пересылаемые между клиентом и сервером.

Для этого он может внедрить ложный ARP-сервер. Поэтому является необходимым как применение дополнительных средств защиты, так и выполнение следующих рекомендаций:

1) физически защитить сервер;
2) защитить критичные файлы системы, например вычислить контрольные суммы на файлы, находящиеся в SYS: LOGIN, SYS: PUBLIC и SYS: SYSTEM, и постоянно производить проверку полученных контрольных сумм;
3) доступ пользователей к ресурсам должен быть организован в соответствии с выбранной политикой безопасности;
4) включить accounting, посредством которого можно регистрировать все попытки произвести login и logout к серверу;
5) стараться не применять процедуру RCONSOLE (удаленной консоли), которая является не только удобным средством, позволяющим осуществлять удаленное управление сервером (включая загрузку и выгрузку), но и вполне доступной мишенью для нарушителя;
6) добавить команду EXIT в System Login Script.

Безопасность в сетях Windows



Из достаточно большого числа уязвимостей ОС Windows, связанных как с непроработанностью вопросов сетевой безопасности, ошибками при проектировании ее сетевой части, так и с некорректным администрированием, наиболее проблематичными являются следующие:

1) отправка атакуемой системе непрерывного потока произвольных UDP-пакетов (UDP flood) по порту 53 (DNS) приводит к краху DNS-сервера (Service Pack 3);
2) атака, направленная на DNS-сервер;
3) атака типа «человек-в-середине».

Кроме того, причиной успешных атак на ОС Windows может стать некорректная реализация выбранной политики безопасности (разрешения отдельным пользователям или группам пользователей при работе с рабочей станцией из ЛВС задаются некорректно). Чтобы избежать этих неприятностей, следует:

1) удалить учетную запись пользователя Guest;
2) ограничить доступ пользователей к рабочей станции из ЛВС путем задания соответствующих прав, например запретить доступ к сети группе Everyone;
3) по возможности не использовать удаленное редактирование системного реестра [2].

Выполнение перечисленных рекомендаций позволит обезопасить работу в ЛВС под управлением ОС Windows.

Интеграция протоколов безопасности в прикладное ПО реализована за счет создания нового интерфейса для приложений Win32 — SSPI (Security Support Provider Interface). Его применение позволит унифицировать обращение к функциональным возможностям данных протоколов и изолировать прикладное ПО от выполнения функций безопасности. Вынесение этих функций за рамки прикладного ПО позволит минимизировать влияние ошибок или закладок в программах на уровень обеспечиваемой безопасности.

В заключение следует отметить, что развитая сетевая поддержка в ОС Windows существенно затрудняет обеспечение безопасности ЛВС. Следовательно, при эксплуатации этой операционной системы целесообразно всемерно упрощать способы и протоколы взаимодействия компьютеров и прикладных систем между собой.

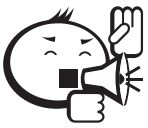
Контрольные вопросы по главе 6

1) Каким образом может осуществляться встраивание программных средств защиты информации в ИТС?
2) Опишите существующую классификацию уровней защиты информации.
3) Какие вы знаете стандарты защиты информации?
4) На основе каких принципов строится современная подсистема информационной безопасности?
5) Охарактеризуйте угрозы информационной безопасности для локальных рабочих станций.
6) Опишите методы и средства обеспечения информационной безопасности локальных рабочих станций.
7) Приведите общую схему построения системы аудита.
8) Перечислите основные угрозы для локальных вычислительных сетей.
9) Какого типа атака возможна в локальной сети Novell NetWare и какие меры для ее предотвращения необходимо принять?
10) Какого типа атаки возможны в сетях Windows NT и какие рекомендации по их предупреждению вы можете предложить?

Глава 7

ВИРУСЫ И УГРОЗЫ, СВЯЗАННЫЕ С ВИРУСАМИ

7.1 Вредоносные программы



По-видимому, наиболее изощренную угрозу для компьютерных систем представляют собой программы, использующие уязвимые места самих систем. В данном контексте рассматриваются не только прикладные программы, но и служебные, такие как редакторы и компиляторы.

Мы начнем с обзора всего спектра программных угроз [4]. На рис. 7.1 показана общая схема классификации программных угроз, или вредоносных программ. Такие программы можно разделить на две категории: нуждающиеся в программе-носителе и независимые программы. К первой категории относится программный код, который не может работать независимо от некоторой реальной прикладной программы, утилиты или системной утилиты. Ко второй категории принадлежат самостоятельные программы, которые могут быть запущены стандартными средствами операционной системы, как любая другая программа.

Кроме того, мы разделим вредоносные программы на обычные и размножающиеся. К первым относятся фрагменты программ, которые активизируются, когда программа-носитель выполняет вполне конкретные действия. Ко второй группе могут относиться как программные фрагменты (вирусы), так и независимые программы («черви», «бактерии»), во время работы создающие свои копии, которые могут выполняться впоследствии в той же или другой системе.

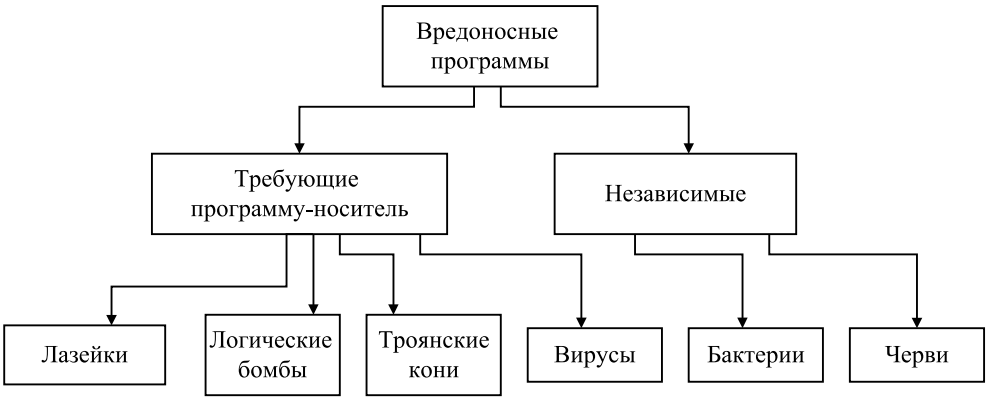


Рис. 7.1 – Классификация вредоносных программ

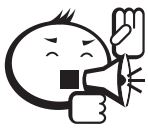
Хотя классификация, приведенная на рис. 7.1, и позволяет систематизировать информацию по этому вопросу, она не дает полного описания реальной картины. В частности, логические бомбы и «троянские кони» могут быть частями вируса или «червя» [4].

7.2 Лазейки

Лазейка — это секретная точка входа в программу, позволяющая тому, кто знает о существовании этой лазейки, получить доступ в обход стандартных процедур защиты.

Лазейки уже много лет совершенно законно используются в программистской практике для ускорения отладки и тестирования программ [4].

Эта возможность обычно применяется программистом при создании приложения, имеющего процедуру аутентификации или слишком длительную программу установки, требующую от пользователя ввода множества различных параметров перед каждым запуском программы. С целью ускорения процесса отладки разработчик может воспользоваться специальным уровнем привилегий или обойти процедуру установки или аутентификации. Для программиста также может быть полезна возможность запустить программу даже тогда, когда по причине каких-то неполадок встроенная процедура аутентификации не позволяет этого сделать.



Лазейка представляет собой программный код, который реагирует на специальную последовательность, введенную с клавиатуры, либо активизируется тогда, когда вводится определенный идентификатор пользователя или в ответ на последовательность каких-то маловероятных событий.

Лазейки представляют собой угрозу, когда они позволяют недобросовестным программистам получить несанкционированный доступ. Именно лазейка была основной причиной угрозы, на которой был построен сюжет фильма «Военные игры» (*War Games*). Реализовать контроль возможных лазеек стандартными средствами операционной системы очень трудно. Меры защиты в данном случае должны быть сфокусированы на контроле процесса разработки программного обеспечения и его обновления.

7.3 Логическая бомба

Одним из самых старых типов вредоносных программ, возникших еще до появления вирусов и «червей», является логическая бомба. Логическая бомба представляет собой программный код, внедренный в какую-то полезную программу, который должен «взорваться» при выполнении определенных условий. Примерами условий, которые запускают логическую бомбу, могут быть присутствие или отсутствие каких-то файлов, наступление определенного дня недели или определенной даты, имя конкретного пользователя, инициировавшего запуск приложения.

Пример

Например, логическая бомба проверяла наличие определенного табельного номера сотрудника (автора бомбы) и срабатывала тогда, когда этот табельный номер отсутствовал в двух идущих подряд ведомостях по начислению зарплаты.

После запуска бомба может изменять или удалять данные или целые файлы, вызывать зависание машины или выполнять какие-то другие разрушительные действия [4].

7.4 «Троянские кони»

«Троянский конь» представляет собой полезную или кажущуюся полезной программу или командную процедуру, содержащую скрытый код, который после запуска программы-носителя выполняет нежелательные или разрушительные функции.

Программа этого типа может служить для опосредованного выполнения операций, которые несанкционированный пользователь не может выполнить непосредственно.

Пример

Например, для получения доступа к файлам другого пользователя на компьютере, находящемся в совместном пользовании нескольких человек, злоумышленник может создать программу «троянского коня», которая в ходе выполнения изменит параметры контроля доступа к файлам соответствующего пользователя, сделав эти файлы открытыми для всех [4].

Создав такую программу, автор может спровоцировать других пользователей запустить ее, поместив эту программу в общедоступный каталог и присвоив ей имя, которое большинству пользователей покажется именем полезной программы или утилиты. Примером может служить программа, якобы создающая список файлов пользователя в нужном формате. После того как какой-нибудь пользователь запустит такую программу, автор программы может получить доступ к информации, содержащейся в файлах этого пользователя.

Пример

Примером «троянского коня», который очень трудно выявить, является компилятор, модифицированный с целью внедрения дополнительного кода в компилируемые программы определенного вида, например в программы входа в систему. Этот код представляет собой лазейку в модуле регистрации, который позволяет автору программы войти в систему с помощью специального пароля. Обнаружить такого «троянского коня» по исходному коду программы входа в систему невозможно.

Вторым источником мотивации для написания «троянского коня» является разрушение данных. В этом случае программа, которая выполняет какие-то полезные функции (например, программа-калькулятор), может без каких бы то ни было внешних проявлений удалить файлы пользователя.

7.5 Вирус

Вирус представляет собой программу, которая может «заражать» другие программы путем их модификации. В модифицированный код включается код вируса, в результате чего код вируса может продолжать заражать другие программы [4].

Биологические вирусы являются небольшими фрагментами генетического кода — ДНК или РНК, — использующими механизм жизнедеятельности живой клет-

ки для создания тысяч абсолютных копий оригинального вируса. Подобно своему биологическому двойнику, компьютерный вирус несет в своем программном коде рецепт создания совершенных копий самого себя. Внесенный в компьютерную систему, типичный вирус временно захватывает управление дисковой операционной системой компьютера. Затем, при каждом контакте зараженного компьютера с незараженным программным обеспечением очередная копия вируса помещается в новую программу. Таким образом инфекция может передаваться от компьютера к компьютеру ничего не подозревающими пользователями, обменивающимися содержимым магнитных дисков или пересылающими программы по сети. Сеть, с ее возможностями доступа к приложениям и системным службам других компьютеров, является прекрасной «питательной средой» для распространения вируса.

7.6 «Черви»

Сетевые программы — «черви» используют сетевые соединения для распространения от одной системы к другой. Во время работы на отдельном компьютере сетевой «червь» может вести себя как компьютерный вирус или «бактерия», либо внедрять «троянских коней», либо выполнять какие-то другие разрушительные или подрывные операции [4].

Для размножения сетевой «червь» использует какое-нибудь из сетевых средств доставки информации. Примерами таких средств могут быть следующие службы.

- Электронная почта. «Червь» отправляет свою копию по почте в другую систему.
- Удаленный вызов программ. «Червь» запускает свою копию на выполнение в другой системе.
- Доступ к удаленной системе. «Червь» входит в удаленную систему как пользователь, а затем использует команду копирования себя из одной системы в другую.

Новая копия программы «червь» в результате оказывается запущенной в удаленной системе, где в дополнение ко всем другим предусмотренным операциям «червь» продолжает размножаться указанным выше способом.

Сетевой «червь» во многом подобен компьютерному вирусу: у него тоже есть инкубационный период, фаза распространения, фаза активизации и фаза выполнения:

В фазе распространения обычно выполняются следующие функции.

1) Поиск других систем, которые можно заразить, путем проверки списков известных данному компьютеру узлов или других подобных объектов, хранящих информацию об адресах удаленных систем.

2) Установление соединения с удаленной системой.
3) Копирование своего кода в удаленную систему и инициирование ее запуска там.

Перед тем как копировать себя в другую систему, сетевой «червь» может также пытаться проверить, не была ли система уже инфицирована ранее. В многозадачной среде он может также скрывать свое присутствие с помощью назначения себе названия, соответствующего системному процессу, или с помощью использования какого-либо другого имени, не вызывающего подозрения у системного оператора.

7.7 Бактерии

Бактерии являются программами, не повреждающими сами по себе никаких файлов. Единственной целью «бактерии» является воспроизведение себе подобных.

Типичная программа-бактерия может просто запустить две собственные копии в многозадачной среде или создать два новых файла, содержащих по копии оригинальной программы «бактерий». Затем каждая из копий может создать еще две копии и т. д. Скорость размножения «бактерий» растет экспоненциально, что в конце концов приводит к быстрому захвату всех ресурсов процессора, памяти или дискового пространства, в результате чего происходит отказ пользователям в доступе к этим ресурсам [4].

7.8 Природа вирусов

Вирусы могут делать все, что могут делать обычные программы. Единственное различие состоит в том, что вирус присоединяется к другой программе и выполняется скрытно в процессе работы программы-носителя. Во время своего выполнения вирус может выполнить любую операцию, например стереть файлы документов и программы [4].

Жизненный цикл типичного вируса состоит из четырех этапов.

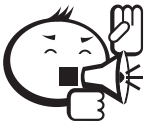
1) Инкубационный период. Вирус никак не проявляется. В конце концов вирус будет активизирован некоторым событием, например наступлением определенной даты, присутствием другой программы или файла, появлением достаточного места на диске. Инкубационный период имеют не все вирусы.
2) Фаза распространения. Вирус помещает свою копию в другие программы или в определенные системные области на диске. Теперь все инфицированные программы будут содержать копию вируса, каждая из которых тоже должна будет когда-нибудь пройти свою фазу распространения.

3) Фаза активизации. Вирус активизируется для выполнения функции, с которой он создавался. Фаза активизации может быть инициирована самыми разными системными событиями, например наличием определенного числа копий данного вируса в системе.

4) Фаза выполнения. Выполняется содержащаяся в вирусе функция. Эта функция может быть как вполне безобидной (например, вывод сообщения на экран), так и совершенно деструктивной (например, уничтожение программ и файлов с данными).

Работа большинства вирусов построена в соответствии с архитектурными принципами конкретной операционной системы и в некоторых случаях даже конкретных аппаратных средств. Таким образом, в их основе лежит использование недостатков и нюансов тех или иных систем.

7.9 Структура вируса



Вирус может помещаться как в начало или конец исполняемой программы, так и встраиваться в нее каким-то иным образом. Главная идея работы вируса состоит в том, чтобы при запуске программы-носителя сначала выполнился код вируса и только затем — код самой программы [4].

Выполнение инфицированной программы начинается с выполнения кода вируса следующим образом. В первой строке кода программы указана команда перехода в начало основного кода вируса. Вторая строка представляет собой специальный маркер, используемый вирусом для проверки того, что программа, выбранная в качестве потенциальной жертвы, не была заражена этим вирусом раньше.

При запуске инфицированной программы управление сразу же передается в основное тело вируса. Программный код вируса сначала ищет незараженные файлы и заражает их. После этого вирус может выполнить какие-то другие действия, обычно враждебные по отношению к системе. Эти действия могут иметь место при каждом запуске инфицированной программы или же только при выполнении определенных условий. Закончив работу, вирус передает управление программе-носителю. Если инфицирование осуществляется достаточно быстро, то пользователь вряд ли сможет заметить при запуске программы, что она инфицирована.

Вирусы, построенные по только что описанному принципу, легко обнаружить, поскольку в результате инфицирования увеличивается длина файла инфицируемой программы. Чтобы обойти столь простые методы обнаружения, вирус должен сжать исполняемый файл, чтобы инфицированная и неинфицированная версии файла оказались одинаковой длины. Когда такая программа запускается на выполнение, управление передается вирусу, который делает следующее.

1) Обнаружив неинфицированный файл, вирус сначала сжимает его, в результате получая новый файл, длина которого меньше исходной на длину кода вируса.

2) Копия вируса присоединяется к началу сжатой программы.

3) Сжатая версия инфицированной оригинальной программы, распаковывается.

4) Распакованная оригинальная программа выполняется.

В рассмотренном выше примере вирус только размножается. Но он может содержать и логическую бомбу, как описано в предыдущем примере.

7.10 Начальное инфицирование

Попав в систему путем заражения некоторой программы, вирус получает возможность при выполнении этой программы заразить некоторые другие или вообще все исполняемые файлы данного компьютера. Таким образом, появления вирусной инфекции можно вообще не допустить, если не позволить вирусу первый раз попасть в систему.

Задача предотвращения вирусного заражения относится к разряду очень сложных, поскольку вирус может оказаться в любой программе, попадающей в систему извне. Таким образом, в уязвимом положении оказывается любой, кто не создал собственную операционную систему и все прикладное программное обеспечение с помощью собственных аппаратных средств.

В большинстве случаев начальное инфицирование происходит через носители информации, с которых зараженные программы копируются в компьютер. Многие из таких программ относятся к разряду игровых или же являются простыми, но удобными утилитами, которыми служащие пользуются сначала на своих домашних компьютерах, а затем приносят на работу. Иногда зараженные программы содержатся даже на упакованных дистрибутивных дисках разработчиков программного обеспечения.

7.11 Типы вирусов

С тех пор как появились вирусы, началась и бесконечная борьба между авторами вирусов и авторами антивирусных программ. Как только вырабатывались эффективные методы противодействия уже известным вирусам, появлялись новые типы вирусов.

В [4] предлагается следующая классификация наиболее важных типов вирусов.

Паразитный вирус. Традиционная и до сих пор самая распространенная форма вируса. Паразитный вирус добавляет свой код к исполняемым файлам и размножается при каждом запуске инфицированной программы, находя другие файлы, которые можно было бы инфицировать.

Резидентный вирус. Размещается в оперативной памяти как часть резидентной системной программы. С момента размещения в памяти инфицирует любую запускаемую программу.

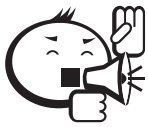
Загрузочный вирус. Инфицирует главную загрузочную запись или загрузочный сектор и распространяется, когда система загружается с зараженного диска.

Вирус-невидимка. Разновидность вируса, имеющего специально предусмотренное свойство, защищающее вирус от обнаружения антивирусным программным обеспечением.

Полиморфный (мимикрирующий) вирус. Вирус, код которого изменяется при каждом новом заражении, что делает практически невозможным обнаружить его по «сигнатуре».

Один из примеров вируса-невидимки (stealth virus) был рассмотрен выше: вирус, использующий сжатие, чтобы длина инфицированного файла была в точности равна длине исходного.

Существуют и другие, гораздо более хитроумные решения. Например, вирус может перехватывать обращения к функциям ввода-вывода и при попытках прочитать подозрительные части диска с помощью этих функций возвращать оригинальные неинфицированные версии программ. Таким образом, применяемая в данном случае характеристика *стелс* (скрытный) относится не столько к вирусам, сколько к технологии, обеспечивающей вирусу защиту от обнаружения.



Полиморфный вирус создает при размножении копии, эквивалентные по функциональности, но существенно различающиеся по двоичному представлению кода. Как и в случае с вирусами-невидимками, это делается с целью противостоять программам, обнаруживающим вирусы. Для таких вариаций представления кода вирус может вставлять в свой код генерируемые случайным образом избыточные команды или же изменять порядок следования не зависящих друг от друга команд.

Более эффективным подходом является шифрование. Часть вируса, называемая *механизмом управления мутациями* (mutation engine), генерирует случайное значение ключа, с помощью которого шифрует остальной код вируса. Ключ сохраняется вместе с вирусом, а механизм управления мутациями видоизменяется. Во время запуска инфицированной программы вирус с помощью сохраненного ключа расшифровывается. При новом инфицировании генерируется новый ключ.

Еще одним оружием в арсенале авторов вирусов является пакет инструментальных средств для разработки вирусов. Такой пакет позволяет даже относительному новичку быстро создать целый набор вирусов разных типов. Хотя вирусы, созданные с помощью пакета инструментальных средств разработки, обычно оказываются менее изощренными в сравнении с вирусами, созданными «с нуля», неограниченное число новых вирусов, которые можно генерировать с помощью пакета, представляет собой достаточно серьезную проблему для схем антивирусной защиты.

7.12 Макровирусы

За последние годы число вирусов, регистрируемых на корпоративных узлах, стремительно возросло. Практически весь этот прирост связан с распространением нового типа вирусов, называемых макровирусами.

Согласно информации NCSA (National Computer Security Agency — Национальное агентство компьютерной безопасности США), которая доступна по адресу www.nsca.com, макровирусы сегодня составляют две трети от общего количества вирусов [4].

Макровирусы особенно опасны по следующим причинам.

1) Макровирусы независимы от платформы. Так, практически все макровирусы поражают документы Microsoft Word. Поэтому любая аппаратно-программная система, поддерживающая Word, может быть заражена таким вирусом.
2) Макровирусы инфицируют документы, а не выполняемый код. А информация, вводимая в компьютерную систему, по большей части представлена в форме документов, а не программ.
3) Макровирусы быстро распространяются. Чаще всего распространение происходит по электронной почте.

Существование макровирусов построено на использовании средств поддержки макросов, предлагаемых в Word и других офисных приложениях (например, Microsoft Excel). По сути, макрос представляет собой программу, встроенную в документ текстового процессора или файл какого-то другого типа.

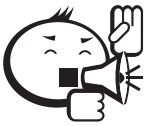
Обычно пользователи используют макросы для того, чтобы автоматизировать выполнение часто выполняемых действий, что позволяет сэкономить время. Язык макросов чаще всего является каким-нибудь вариантом языка программирования Basic. Пользователь может записать последовательность нажатий клавиш в виде макроса, а затем настроить программу так, чтобы записанный макрос вызывался нажатием функциональной клавиши или какой-то специальной комбинацией клавиш.

Создание макровирусов оказывается возможным благодаря существованию автоматических макросов. Это макросы, которые выполняются автоматически, без явной активизации их пользователем. Типичными автоматически происходящими событиями являются открытие, закрытие файла, а также запуск приложения. Во время своего выполнения макрос может копировать себя в другой документ, удалять файлы и выполнять любые другие действия, разрушающие систему пользователя.

Обычно распространение макровируса происходит следующим образом. Автомакрос или командный макрос вставляется в документ Word, который передается в компьютерную систему по электронной почте или с внешнего носителя информации. В какой-то момент после открытия документа макровирус начинает выполняться. Он копирует свой код в глобальный файл макросов. При следующем запуске Word становится активным инфицированный глобальный файл макросов. При выполнении макрос может размножаться и выполнять действия, наносящие вред системе.

В современные версии Word встроена защита от макровирусов. Ведущие разработчики антивирусных программ тоже создали средства для обнаружения и удаления опасных макровирусов. Однако, как и в случае любых других вирусов, «гонка вооружений» в области макровирусов продолжается.

7.13 Антивирусная защита



Идеальным решением проблемы вирусов является предотвращение инфицирования: не следует допускать начального проникновения вируса в компьютерную систему. Этой цели в общем достичь невозможно, хотя предпринятые превентивные меры могут снизить число успешно завершенных вирусами атак.

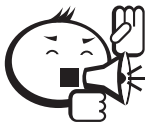
Почти идеальный подход должен обеспечивать выполнение следующих требований [4].

- Обнаружение. Если заражение произошло, оно должно быть немедленно обнаружено с установлением места обитания вируса.
- Идентификация. Как только заражение вирусом обнаружено, необходимо идентифицировать тип вируса, инфицировавшего программу.
- Удаление. Как только вирус идентифицирован, следует удалить все следы вируса из инфицированных программ и восстановить программы в их ис-

ходный вид. Важно удалить вирус из всех инфицированных систем, чтобы болезнь не распространялась дальше.

Если вирус обнаружен, но его не удается идентифицировать или удалить из системы, альтернативой является удаление инфицированной программы с последующей ее новой загрузкой с резервной копии.

Технологии разработки вирусов и антивирусов идут рука об руку. Первые вирусы представляли собой сравнительно простые фрагменты кода и могли быть удалены с помощью относительно простых антивирусных программ. По мере усложнения вирусов антивирусное программное обеспечение тоже становилось все сложнее и изощреннее.



Антивирусные программы разделяются на четыре поколения [4]:

- Первое поколение: обычные сканеры.
- Второе поколение: эвристические анализаторы.
- Третье поколение: мониторы.
- Четвертое поколение: полнофункциональные системы защиты.

Антивирусные программы-сканеры первого поколения для идентификации вирусов использовали характерные для соответствующих вирусов сигнатуры. Вирусы могли содержать «групповые символы», но все копии вируса имели в основном одну и ту же структуру и неизменный код. Такие программы-сканеры, использующие сигнатуры, могли обнаруживать только известные вирусы. Другой тип сканеров первого поколения предполагал поиск несоответствий текущих значений длин файлов со значениями, сохраненными в специальной базе данных.

Сканеры второго поколения уже не ориентированы на конкретные сигнатуры. Вместо этого, в них начали применять эвристический анализ, с помощью которого можно было сделать вывод о возможном наличии в программе вируса. Одна из разновидностей таких сканеров предполагала поиск в программе фрагментов кода, характерного для вирусов. Например, сканер мог искать начало цикла шифрования, используемого полиморфным вирусом, и пытаться открыть ключ шифрования. Получив ключ, сканер мог расшифровать тело вируса, идентифицировать вирус, удалить его из программы и вернуть программу в рабочее состояние.

Другим подходом, применявшимся в антивирусных программах второго поколения, была проверка целостности. С каждой программой можно связать контрольную сумму. Если вирус инфицирует программу, не меняя при этом контрольной суммы, то проверка целостности обязательно это обнаружит. Чтобы противостоять вирусам, которые при заражении могут менять соответствующую контрольную сумму, можно использовать некоторую функцию хэширования с шифрованием. Ключ шифра хранится отдельно от программы, чтобы вирус не мог сгенерировать новый хэш-код и зашифровать его. Использование функции хэширования с шифрованием вместо обычной контрольной не дает вирусу возможности модифицировать программу таким образом, чтобы результат хэширования после инфицирования не изменялся.

Программы третьего поколения представляли собой резидентные программы, выявляющие вирусы по выполняемым ими действиям, а не по их структуре в инфицированной программе. Преимуществом таких программ было то, что для них не требовалось постоянно обновлять базу данных сигнатур и эвристик для все большего числа вирусов. Вместо этого достаточно было определить относительно небольшой набор действий, характеризующих возможные проявления вируса.

Продукты четвертого поколения представляют собой пакеты, объединяющие в единое целое все существующие антивирусные технологии. Такой подход, помимо выполнения сканирования и наличия компонентов, позволяющих регистрировать определенные действия вирусов, предполагает наличие средств управления доступом, с помощью которых можно ограничить возможности вирусов по проникновению в систему и по внесению изменений в файлы с целью распространения инфекции под видом обновления.

Вирусная «гонка вооружений» продолжается. С появлением пакетов четвертого поколения появилась возможность построения всеобъемлющей стратегии антивирусной защиты, являющейся органической частью общих мероприятий по обеспечению защиты компьютерной системы.

7.14 Перспективные методы антивирусной защиты

В результате постоянной разработки все более совершенных подходов к осуществлению антивирусной защиты появляются новые и более надежные продукты. В данном разделе мы остановимся на двух наиболее важных направлениях развития данной области защиты систем [4].

Полное декодирование

Технология полного декодирования (GD — general decryption) позволяет антивирусной программе легко обнаруживать даже самые сложные полиморфные вирусы, сохраняя при этом высокую скорость сканирования.

Напомним, что при выполнении содержащего полиморфный вирус файла перед активизацией вирус должен быть расшифрован. Для выявления подобных структур выполняемый файл проходит через GD-сканер, состоящий из следующих элементов:

- Эмулятор процессора. Представляет собой программную реализацию виртуального компьютера. Команды, имеющиеся в выполняемом файле, интерпретируются эмулятором вместо того, чтобы выполняться реальным процессором. Эмулятор содержит программные версии всех регистров и других аппаратных средств процессора, так что сам процессор оказывается вне зоны действия программ, интерпретируемых эмулятором.

- Сканер сигнатур вирусов. Модуль, выполняющий сканирование проверяемого кода на предмет выявления в нем сигнатур известных вирусов.
- Модуль управления эмуляцией. Управляет выполнением проверяемого кода.

Загрузив программу, эмулятор начинает интерпретировать одну за одной команды проверяемого кода. Если в программе содержится процедура дешифрования тела вируса, ее код тоже будет интерпретирован. В результате вирус сам откроет себя антивирусной программе. Периодически модуль управления прерывает ход эмуляции, чтобы выполнить сканирование полученного кода на предмет наличия в нем сигнатур вирусов.

Во время интерпретации код вируса не может нанести реального вреда компьютеру, так как код выполняется в изолированной и контролируемой виртуальной среде.

Самым сложным вопросом реализации GD-сканеров является определение того, как долго нужно выполнять интерпретацию проверяемой программы. Обычно вирус активизируется вскоре после запуска программы, но это правило не является догмой. Чем дольше эмулятор проверяет программу, тем выше вероятность обнаружения скрытых вирусов. Однако неоправданно большое время проверки и высокие требования к потребляемым ресурсам, очевидно, не очень удовлетворяют пользователей.

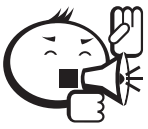
Цифровая иммунная система

Цифровая иммунная система представляет собой сложную технологию антивирусной защиты, разработанную компанией IBM. Причиной разработки послужила возрастающая угроза распространения вирусов через Internet. Сначала мы скажем несколько слов о самой угрозе, а затем перейдем к описанию подхода, предложенного IBM.

В настоящее время угроза вирусной инфекции характеризуется относительно низкой частотой появления новых вирусов и новых мутаций старых вирусов. Антивирусное программное обеспечение обновляется, как правило, ежемесячно, и этого оказывается достаточным для того, чтобы держать проблему под контролем. Однако в ближайшие годы скорость распространения вирусов может существенно возрасти под влиянием следующих тенденций развития технологий Internet:

- Интегрированные почтовые системы. Системы типа Lotus Notes и Microsoft Outlook допускают пересылку чего угодно кому угодно.
- Мобильные программы. Технологии, подобные Java и ActiveX, позволяют программам самостоятельно перемещаться из одной системы в другую.

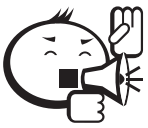
В ответ на угрозу распространения вирусов, которую потенциально несут эти возможности Internet, IBM разработала прототип цифровой иммунной системы. Эта система развивает технологию эмуляции, о которой говорилось в предыдущем разделе, предлагая комплекс эмулятора общего назначения и системы обнаружения вирусов.



Целью цифровой иммунной системы является создание схемы быстрого реагирования, позволяющей регистрировать вирусы практически в момент их появления. Когда новый вирус появляется в вычислительной сети организации, иммунная система находит его, анализирует, добавляет необходимые средства обнаружения этого вируса и защиты от него, удаляет вирус и передает информацию о попытке проникновения вируса системам AntiVirus компании IBM, чтобы вирус и в других местах мог быть выявлен до того, как он начнет выполняться.

Ниже излагается последовательность действий цифровой иммунной системы:

1) Специальная программа, работающая на каждом персональном компьютере, выполняет мониторинг, используя различные методы эвристического анализа поведения системы и подозрительных изменений в программах, а также информацию о сигнатурах известных вирусов. Обнаружив подозрительные действия, программа мониторинга отправляет копию подозрительной на предмет заражения программы машине-администратору сети организации.
2) Машина-администратор шифрует полученный образец и отправляет его центральной машине-анализатору, выполняющей анализ вирусов.
3) Машина-анализатор создает виртуальную среду, в которой можно выполнить безопасный запуск инфицированной программы для анализа. Для этого может применяться технология программной эмуляции или создание какой-то иной защищенной среды, в которой подозреваемая программа может быть выполнена и изучена. Обнаружив вирус, машина-анализатор генерирует рекомендации, касающиеся вопросов идентификации и удаления вируса.
4) Созданные рекомендации передаются обратно машине-администратору.
5) Машина-администратор пересылает рекомендации инфицированной машине-клиенту.
6) Рекомендации рассылаются также всем другим машинам-клиентам организации.
7) Все другие подписчики системы также получают регулярные обновления антивирусных пакетов, что позволяет защититься от новых вирусов.



Успех цифровой иммунной системы зависит от способности машины-анализатора выявлять новые вирусы и их модифицированные штаммы. С помощью постоянного анализа и мониторинга всех появляющихся вирусов можно обеспечить постоянное обновление программного обеспечения цифровой иммунной системы с целью организации надежной защиты от угрозы вирусной инфекции.

Контрольные вопросы по главе 7

1) Приведите классификацию вредоносных программ.
2) Охарактеризуйте понятие лазейки.
3) Опишите логическую бомбу.
4) Охарактеризуйте понятие «троянского коня».
5) Охарактеризуйте понятие вируса.
6) Опишите понятие «червя».
7) Охарактеризуйте функции бактерии.
8) Опишите природу вирусов.
9) Охарактеризуйте структуру вирусов.
10) Опишите типы вирусов.
11) Макровирусы.
12) Опишите антивирусные программы первого поколения.
13) Опишите антивирусные программы второго поколения.
14) Опишите антивирусные программы третьего поколения.
15) Опишите антивирусные программы четвертого поколения.
16) Технология полного декодирования.
17) Цифровая иммунная система.

Глава 8

БРАНДМАУЭРЫ

Брандмауэры могут быть эффективным средством защиты локальной системы или компьютерной сети от угроз, имеющих сетевую природу, в то же время не ограничивающим связь с внешним миром через глобальные сети и Internet.

В главе излагаются функциональные возможности брандмауэров и принципы, на основе которых выполняется их проектирование. Рассматриваются вопросы защиты самих брандмауэров и обсуждается концепция высоконадежной системы (т. е. системы, заслуживающей доверия с точки зрения обеспечения защиты данных) или, другими словами, защищенной операционной системы.

8.1 Принципы разработки брандмауэров

Информационные системы корпораций, правительственных учреждений и других организаций постоянно развиваются в следующих направлениях [4]:

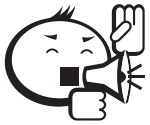
- Система централизованной обработки данных, работающая на базе мэйнфрейма, к которому непосредственно подключено некоторое число терминалов.
- Локальная сеть, объединяющая персональные компьютеры и терминалы друг с другом и мэйнфреймом.
- Объединенная сеть, состоящая из нескольких локальных сетей, связанных друг с другом персональных компьютеров, серверов и, возможно, одного или двух мэйнфреймов.
- Сеть масштаба предприятия, состоящая из нескольких находящихся на достаточно большом расстоянии друг от друга объединенных сетей, связываемых между собой с помощью ведомственной глобальной сети.

- Связь через Internet, при которой различные объединенные сети оказываются подключенными к Internet и могут быть также связанными друг с другом ведомственной глобальной сетью.

Сегодня связь с Internet для большинства организаций уже является привычным делом. Для многих организаций информация и услуги Internet жизненно необходимы. Кроме того, доступ к Internet требуется и многим индивидуальным пользователям, и если их локальная сеть не обеспечивает такого доступа, они самостоятельно используют средства удаленного доступа для подключения своих персональных компьютеров к Internet через поставщика услуг Internet (провайдера). Но хотя доступ к Internet и дает организации очевидные преимущества, такой доступ в то же время предоставляет возможность доступа к ресурсам внутренней сети организации и внешнему миру. Ясно, что это несет в себе определенную угрозу для организации. Для защиты от этой угрозы можно оборудовать каждую рабочую станцию и каждый сервер внутренней сети надежными средствами защиты типа системы защиты от вторжений, но такой подход, очевидно, непрактичен.

Рассмотрим, например, сеть с сотнями или даже тысячами отдельных систем, на которых работают самые разные версии UNIX и Windows. Если будет выявлен какой-либо изъян в системе защиты, придется внести изменения в средства защиты всех систем, которые потенциально могут оказаться уязвимыми. Альтернативой, которая становится все более популярной, является использование брандмауэра. Это устройство размещается между внутренней сетью организации и Internet, обеспечивая контролируемую связь и воздвигая внешнюю стену, или границу. Целью создания такой границы является защита внутренней сети от несанкционированного проникновения извне через Internet и организация единого центра, в котором должны применяться средства защиты и контроля. Брандмауэр может представлять собой как отдельный компьютер сети, так и группу специально выделенных компьютеров, которые осуществляют функции брандмауэра сети, взаимодействуя между собой.

8.2 Характеристики брандмауэров



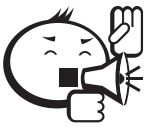
В [4] приводится следующий список основных задач, стоящих перед разработчиками брандмауэров.

1) Весь поток данных, направляемых из внутренней сети во внешний мир, как и идущих в обратном направлении, должен проходить через брандмауэр. Это достигается путем физического блокирования любого доступа к локальной сети в обход брандмауэра. Как будет показано ниже, эта задача может иметь несколько различных конструктивных решений.
2) Из всего потока поступающих к брандмауэру данных пройти брандмауэр должно быть позволено только данным, подвергшимся аутентификации в соответствии с локальной политикой защиты. Как будет показано ниже, в брандмауэрах разных типов могут быть реализованы различные типы политик защиты.

3) Брандмауэр сам по себе должен быть недоступен для вторжения извне. Это предполагает наличие высоконадежной системы с защищенной операционной системой.

В [4] перечислены четыре основных средства, с помощью которых брандмауэры осуществляют контроль доступа и обеспечивают реализацию политик защиты соответствующей вычислительной системы. Изначально брандмауэры осуществляли в основном управление сервисами, но сегодня на них возложены и остальные задачи.

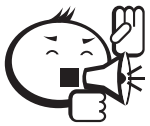
- *Управление сервисами.* Определяет типы служб Internet, к которым можно получать доступ из внутренней сети наружу и из внешнего окружения вовнутрь. Брандмауэр может фильтровать поток данных на основе IP-адресов и номеров портов TCP, может предлагать такой компонент, как прокси-сервер, через который может проходить каждый запрос сервиса и который принимает решение о том, пропустить данный запрос или нет, и, наконец, может содержать и серверные компоненты, такие как Web-сервер или почтовая служба.
- *Управление направлением движения.* Определение направления, в котором могут инициироваться и проходить через брандмауэр запросы к тем или иным службам.
- *Управление пользователями.* Предоставление доступа к службам в зависимости от прав доступа пользователей, обращающихся к этим службам. Эта функция обычно применяется к пользователям внутри сети, защищаемой с помощью брандмауэра (локальным пользователям). Однако она может применяться и к потоку данных, поступающему от внешних пользователей, но это требует реализации в какой-то форме технологии аутентификации, например обеспечиваемой протоколом IPSec.
- *Управление поведением.* Контроль за использованием отдельных служб. Например, брандмауэр может фильтровать электронную почту, отсеивая «спам», или же разрешать доступ извне только к определенной части информации, находящейся на локальном Web-сервере.



Перед рассмотрением особенностей конфигурации брандмауэров различных типов необходимо определить требования, выдвигаемые к брандмауэру. Брандмауэр может предлагать следующие возможности.

1) Брандмауэр представляет собой единственную точку входа, в которой предотвращается несанкционированный доступ внешних пользователей к защищаемой сети, запрещается потенциально уязвимым службам доступ к внутренней сети извне или отправка данных изнутри во внешний мир, также обеспечивается защита от различных атак вторжения с помощью изменения маршрута или указания ложных IP-адресов.

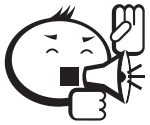
2) Брандмауэр является местом, где осуществляется мониторинг событий, имеющих отношение к защите сети. Для этого в брандмауэре могут быть предусмотрены компоненты контроля и извещения.
3) Брандмауэр является удобной платформой для ряда функций Internet, имеющих непосредственное отношение к безопасности. К таким функциям можно отнести транслятор сетевых адресов, преобразующий локальные адреса в адреса Internet, а также средства управления сетью, осуществляющие контроль за использованием Internet или ведение соответствующего журнала регистрации.
4) Брандмауэр может служить платформой для IPSec. Возможности туннельного режима такого брандмауэра могут применяться при построении виртуальных частных сетей.



Брандмауэры имеют свои ограничения, описанные ниже.

1) Брандмауэр не может защитить от атак, идущих в обход брандмауэра.
2) Брандмауэр не может защитить от внутренних угроз безопасности, например со стороны сотрудника, вступившего в сговор с внешним нарушителем.
3) Брандмауэр не может защитить от угрозы передачи инфицированных вирусами программ или файлов.

8.3 Типы брандмауэров



Рассмотрим три типа брандмауэров: брандмауэр, пакетный фильтр, шлюзы уровня приложений, уровня коммутации (рис. 8.1) [4].

Фильтрующий маршрутизатор

Фильтрующий маршрутизатор (packet-filtering router) принимает решение о том, передавать по сети поступивший пакет IP дальше или отвергнуть его, на основе определенного набора правил (рис. 8.1(а)). Обычно маршрутизатор настраивается таким образом, чтобы фильтровать пакеты, проходящие в обоих направлениях (т. е. как в во внутреннюю сеть, так и из внутренней сети во внешний мир). Правила фильтрования основываются на значениях полей заголовка IP, заголовка транспортного уровня (задающего транспортный протокол), а также номера порта TPC или UDP (определяющего приложение, например SNMP или TELNET).

Пакетный фильтр обычно представлен в виде списка правил, использующих значения полей заголовка IP или TCP. Если обнаруживается соответствие одному из правил, то это правило служит критерием для принятия решения о том, можно

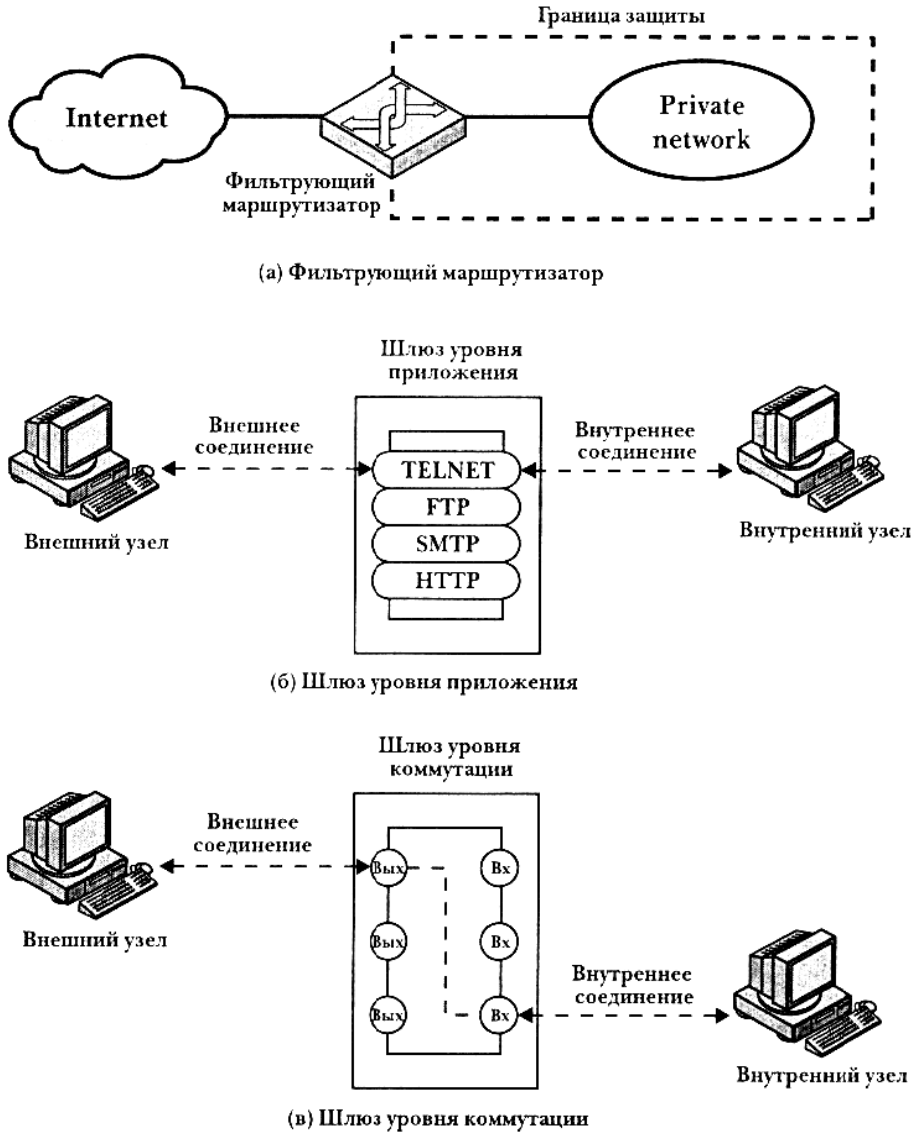


Рис. 8.1 – Типы брандмауэров

ли передать пакет дальше или же его следует отвергнуть. В отсутствие соответствия любому из правил выполняется операция, предусмотренная для использования по умолчанию. Для этого имеются две следующие возможности.

- Default = discard. Все, что не разрешено, запрещено.
- Default = forward. Все, что не запрещено, разрешено.

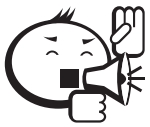
Очевидно, что первая политика соответствует более консервативному подходу. Изначально все оказывается запрещенным, и добавление конкретных служб осуществляется для каждой из них в отдельности. Эта политика более очевидна для пользователей, по отношению к которым брандмауэр выступает как препятствие. Вторая политика в значительной степени облегчает работу конечных пользователей, но зато обеспечивает более низкий уровень защиты, поскольку в данном случае администратору системы защиты приходится принимать меры по каждой новой обнаруженной угрозе безопасности.

Одно из достоинств фильтрующего маршрутизатора связано с его простотой, Кроме того, пакетные фильтры, как правило, остаются незаметными для пользователей и обеспечивают высокую скорость работы. Основными недостатками являются сложность верного выбора правил фильтрации и отсутствие средств аутентификации.

Перечислим некоторые из типов атак, которые могут предприниматься в отношении нарушения системы защиты фильтрующего маршрутизатора, а также соответствующие контрмеры.

- *Подмена IP-адреса.* Нарушитель передает извне пакеты, в которых поле отправителя содержит IP-адрес внутреннего узла сети. Нарушитель надеется, что использование такого адреса даст возможность проникнуть в систему. Контрмерой является отторжение пакетов с внутренними адресами источника, поступивших извне.
- *Использование маршрутизации от источника.* Отправитель указывает маршрут, по которому пакет должен пересылаться по Internet, в надежде на то, что удастся обойти средства защиты, не анализирующие информацию о маршрутизации, задаваемую отправителем. Контрмера — запрет прохождения всех пакетов, использующих данную возможность.
- *Разделение на мелкие фрагменты.* Нарушитель использует опцию фрагментации IP-пакетов для создания фрагментов исключительно малой длины, чтобы информация заголовка TCP попала в отдельный фрагмент. Этот тип атаки ставит своей задачей обойти правила фильтрации пакетов, использующие информацию заголовка TCP. Нарушитель надеется, что фильтрующим маршрутизатором будет просмотрен только первый пакет, остальные фрагменты будут пропущены. Контрмера — отторжение всех пакетов, в которых указан протокол TCP, а значение параметра Fragment Offset протокола IP равно 1.

Шлюз уровня приложения



Шлюз уровня приложения, который также часто называют прокси-сервер (proxy server), работает как ретранслятор данных уровня приложения (рис. 8.1(б)).

Пользователь связывается с таким шлюзом с помощью построенного на основе TCP/IP приложения, такого как Telnet или FTP, а шлюз запрашивает у этого пользователя имя удаленного узла, к которому необходимо получить доступ. Если пользователь отвечает на этот запрос, сообщая правильный идентификатор пользователя и информацию, необходимую для аутентификации, то шлюз связывается с соответствующим приложением на удаленном узле и ретранслирует сегменты TCP, содержащие данные приложения. Если в шлюзе не реализована поддержка прокси-сервером того или иного приложения, соответствующий сервис не предоставляется и данные соответствующего типа не могут пройти через брандмауэр. Более того, шлюз можно настроить, чтобы он поддерживал только определен-

ные возможности приложения, которые администратор сети считает приемлемыми с точки зрения обеспечения защиты.

В общем случае шлюзы уровня приложения обеспечивают более надежную защиту, чем фильтры пакетов. Вместо того чтобы проверять многочисленные возможные комбинации разрешений и запретов на уровне TCP и IP, шлюз уровня приложения обеспечивает работу лишь ограниченного числа приложений. Кроме того, при этом легко протоколировать и контролировать весь входящий поток данных на уровне приложения.

Основным недостатком шлюзов данного типа является дополнительная нагрузка на процессор, которую требует каждое соединение. На самом деле между двумя конечными пользователями устанавливается два связанных соединения и шлюзу приходится проверять весь поток данных в обоих направлениях и переправлять его дальше.

Шлюз уровня коммутации

Третьим типом брандмауэров являются *шлюзы* уровня коммутации (рис. 8.1(в)). Данный шлюз может быть реализован как в виде отдельной компьютерной системы, так и в виде специальной функции шлюза уровня приложения, предлагаемой для некоторых приложений.

Шлюзы уровня коммутации не разрешают внешним узлам устанавливать сквозные TCP-соединения с узлами внутренней сети, а вместо этого устанавливают два TCP-соединения: одно между самим шлюзом и внутренним узлом, а второе — между шлюзом и внешним узлом. После того как оба соединения установлены, шлюз обычно ретранслирует сегменты TCP от одного соединения к другому, не проверяя их содержимого. Защита реализуется путем разрешения или запрета тех или иных соединений.

Типичным случаем применения шлюзов уровня коммутации является ситуация, когда системный администратор доверяет внутренним пользователям. Шлюз можно настроить таким образом, чтобы он поддерживал работу на уровне приложений или функции прокси-сервера для входящих соединений и функции шлюза уровня коммутации — для исходящих. В такой конфигурации шлюз сталкивается с высокими дополнительными нагрузками, связанными с необходимостью проверки входящих данных приложений на предмет выполнения ими запрещенных функций, однако такой нагрузки для исходящих данных можно избежать.

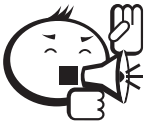
Бастионный узел

Бастионный узел является системой, выбранной администратором брандмауэра в качестве критически важной точки в схеме защиты сети. Как правило, бастионный *узел служит* платформой для шлюза уровня приложения или шлюза уровня коммутации.

Бастионный узел имеет следующие характеристики:

- На аппаратных средствах бастионного узла выполняется защищенная версия операционной системы, в результате чего бастионный узел оказывается высоконадежной системой (trusted system).
- На бастионном узле устанавливаются только те службы, которые сочтет необходимыми сетевой администратор. К таким службам относятся приложения-агенты (прокси-службы) типа Telnet, DNS, FTP, SMTP и средства аутентификации пользователей.
- Бастионный узел может требовать дополнительной аутентификации, прежде чем предоставить пользователю доступ к прокси-службам. Кроме того, каждая прокси-служба может выдать запрос на дополнительную аутентификацию, прежде чем предоставить пользователю доступ.
- Каждая прокси-служба настроена для поддержки только определенного набора команд из общего стандартного множества команд, поддерживаемых приложением.
- Каждая прокси-служба настроена таким образом, чтобы предоставлять доступ только к определенным узлам. Это значит, что вышеупомянутый ограниченный набор команд и возможностей оказывается применимым только к определенной части систем защищаемой сети.
- Каждая прокси-служба обеспечивает широкие возможности для контроля, регистрируя весь поток данных, каждое соединение и длительность использования каждого соединения. Контрольный журнал является исключительно важным средством выявления попыток вторжения.
- Каждый прокси-модуль представляет собой небольшой программный пакет, специально созданный для защиты сети. Ввиду относительной простоты эти модули легко проверить с целью убедиться в отсутствии возможных изъянов в программном коде с точки зрения защиты.
- Каждый прокси-модуль не зависит от других прокси-модулей бастионного узла. Если возникает проблема с работой какого-либо модуля или в нем выявляется изъян защиты, такой модуль может быть удален без каких-либо последствий для других прокси-приложений. Кроме того, если пользователям потребуется доступ к новой службе, администратор сети может просто установить требуемый прокси-модуль на бастионный узел.
- Прокси-модуль обычно не предоставляет никакого доступа к диску, кроме чтения своего файла исходной конфигурации. Это делает задачу внедрения«троянского коня» и других опасных файлов на бастионном узле очень трудной для нарушителя.
- Каждый прокси-модуль работает как непривилегированный пользователь в личном и защищенном каталоге бастионного узла.

8.4 Конфигурации брандмауэров



Помимо конфигурации, когда брандмауэр представлен отдельной системой типа фильтрующего маршрутизатора или шлюзом (рис. 8.1), возможны и более сложные конфигурации, и такие более сложные конфигурации на самом деле используются чаще всего.

На рис. 8.2 показаны три чаще всего встречающиеся конфигурации брандмауэров [4].

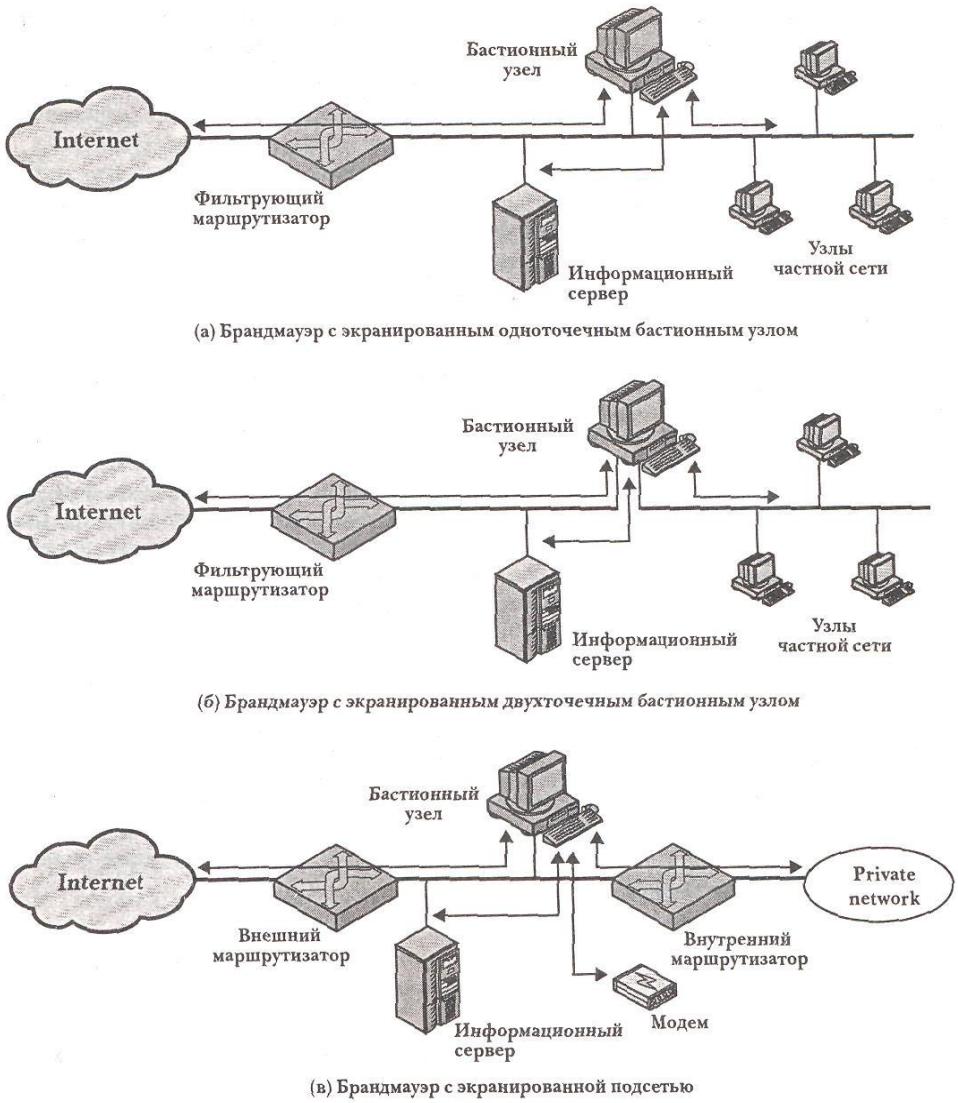


Рис. 8.2 – Конфигурации брандмауэров

В конфигурации брандмауэра с экранированным одноточечным бастионным узлом брандмауэр состоит из двух систем: фильтрующего маршрутизатора и бастионного узла.

Как правило, маршрутизатор при этом настраивается следующим образом.

1) Из потока данных, поступающего из Internet, пропускаются только IP-пакеты, предназначенные для бастионного узла.
2) Из потока данных, поступающего из внутренней сети, пропускаются только пакеты, исходящие из бастионного узла.

Бастионный узел выполняет задачи аутентификации и функции прокси-сервера. Эта конфигурация обеспечивает гораздо более высокий уровень защиты, чем просто фильтрующий маршрутизатор или шлюз уровня приложения по следующим причинам. Во-первых, в данной конфигурации реализована фильтрация как на пакетном уровне, так и на уровне приложения, что обеспечивает достаточную гибкость в выборе политики защиты. Во-вторых, нарушителю для вторжения во внутреннюю сеть приходится преодолевать защиту двух отдельных систем.

Данная конфигурация оказывается достаточно гибкой в плане прямого доступа к Internet. Например, внутренняя сеть может содержать общедоступный информационный сервер типа Web-сервера, для которого не требуется высокий уровень защиты. В таком случае маршрутизатор можно настроить так, чтобы он позволял прямой доступ к данному серверу из Internet.

Если фильтрующий маршрутизатор в только что описанной конфигурации с одноточечным бастионным узлом будет скомпрометирован (т. е. его защита будет взломана), поток данных из Internet может пойти прямо через маршрутизатор в узлы внутренней частной сети. Брандмауэр с экранированным двухточечным бастионным узлом сконфигурирован так, чтобы физически исключить возможность появления такой бреши в системе зашиты. Все преимущества двухуровневой защиты, которыми обладает описанная выше конфигурация, остаются при этом в силе. Опять же, информационный сервер и некоторые другие узлы могут быть подключены к маршрутизатору напрямую, если это согласуется принятой политикой защиты.

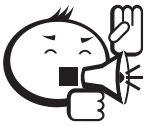
Конфигурация брандмауэра с экранированной подсетью из всех рассматриваемых здесь вариантов обеспечивает самую надежную защиту.

В этой конфигурации установлено два фильтрующих маршрутизатора: один между бастионным узлом и Internet, а второй — между бастионным узлом и внутренней сетью. Данная конфигурация создает изолированную подсеть, которая может состоять только из бастионного узла, но может также иметь и один или несколько информационных серверов, а также модемы, с помощью которых осуществляется удаленный доступ к сети. Обычно к узлам, входящим в состав экранированной подсети, можно получить доступ как из Internet, так и из внутренней

сети, однако поток данных сквозь подсеть блокируется. Данная конфигурация обладает следующими преимуществами.

- Предусмотрено три уровня защиты от вторжения нарушителей.
- Внешний маршрутизатор объявляет в Internet только о существовании экранированной подсети, поэтому внутренняя сеть остается для Internet невидимой.
- Точно так же внутренний маршрутизатор сообщает узлам внутренней сети только о существовании экранированной подсети, поэтому системы внутренней сети не имеют возможности проложить прямые маршруты в Internet.

8.5 Высоконадежные системы



Один из способов расширения возможностей системы противостоять вторжениям нарушителей и вредоносных программ заключается в реализации технологии высоконадежных систем.

Следующий раздел содержит краткий обзор соответствующего предмета. Мы начнем с рассмотрения некоторых базовых концепций управления доступом к данным [4].

Управление доступом к данным

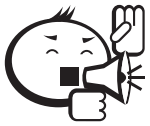
Вслед за получением доступа к системе пользователь получает доступ к одному или нескольким узлам и приложениям. Как правило, такое положение дел не согласуется с требованиями безопасности для тех систем, в которых хранятся важные данные. В ходе выполнения процедуры доступа к системе пользователь может быть идентифицирован. Для каждого пользователя может иметься соответствующий файл профиля, в котором содержится список всех допустимых для него действий и разрешений доступа к файлам. Операционная система может на основе файла профиля ввести соответствующие правила работы.

Но система управления доступом к базам данных должна контролировать доступ не только к файлам баз данных, но и к отдельным записям и даже к отдельным полям. Например, список сотрудников компании может быть доступным для любого работника административного отдела, но только некоторым из них должен быть разрешен доступ к информации о зарплатах сотрудников. Кроме того, здесь может быть и несколько уровней детализации. Если доступ к файлам и использование приложений может разрешить операционная система, после чего уже не проводится никаких проверок на безопасность, то система управления базами данных должна принимать решение о разрешении или запрещении доступа при каждой попытке доступа к данным. Такое решение должно зависеть не только от идентификатора пользователя, но и от того, какие данные и какие части данных запрашиваются, и даже от того, какие данные были открыты данному пользователю раньше.

Общая модель управления доступом, реализуемая на уровне файловой системы или системы управления базами данных, представляется в виде матрицы доступа. Основные элементы данной модели перечислены ниже.

- *Субъект.* Сущность, которая может получать доступ к объектам. В общем случае понятие субъекта отождествляется с понятием процесса. Любой пользователь или приложение получают доступ к объекту с помощью процесса, представляющего этого пользователя или соответствующее приложение.
- *Объект.* Сущность, доступ к которой контролируется. Примерами являются файлы, их части, программы и сегменты памяти.
- *Право доступа.* Способ доступа субъекта к объекту. Примерами являются чтение, запись или выполнение.

По одной оси матрицы представлены идентифицированные субъекты, которые могут пытаться получить доступ к данным. Обычно список состоит из отдельных пользователей или пользовательских групп, хотя можно также контролировать доступ терминалов, узлов или приложений. На другой оси отображены объекты, к которым субъекты могут получать доступ. При самом высоком уровне детализации объектами могут быть отдельные поля базы данных. Более общие группы (т. е. записи, файлы или же вся база данных в целом) тоже могут быть объектами, представленными в матрице. Каждый элемент матрицы указывает права доступа соответствующего субъекта к соответствующему объекту.



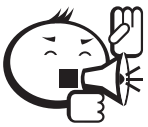
На практике матрица доступа обычно является почти пустой и реализуется в виде декомпозиции одним из следующих двух методов. Матрица может быть разделена на столбцы, в результате чего получаются *списки управления досту пом.* Для каждого объекта в списке управления доступом приводятся пользователи и их права доступа к данному объекту.

В результате декомпозиции по строкам создаются мандаты возможностей. Мандат возможностей определяет для данного пользователя объекты и операции, на использование которых пользователь имеет разрешение. Каждый пользователь получает несколько мандатов возможностей, а кроме того, пользователь может обладать привилегией на использование мандатов других пользователей и предоставлять другим пользователям право использовать свои.

Поскольку мандаты могут быть распределены по всей системе, задача их защиты представляет собой гораздо более сложную задачу, чем задача защиты списков управления доступом. В частности, необходимо предотвратить возможность фальсификации мандатов. Одним из способов решения задачи защиты мандатов является возложение задачи их хранения на операционную систему, которая может поместить их в область, недоступную пользователям.

Идея использования высоконадежных систем

Многие вопросы, которые мы обсуждали ранее, были связаны с проблемой защиты отдельного сообщения или объекта от пассивных или активных атак отдельного пользователя. Однако существует еще одна задача с широкой областью применения, заключающаяся в защите данных или ресурсов на основе уровней безопасности. Эта задача часто встречается в военном деле, где информация обычно классифицируется как открытая (U — unclassified), для служебного пользования (C — confidential), секретная (S — secret), совершенно секретная (TS — top secret). Данный подход применяется и в других областях, где информацию можно разбить на большие категории, а пользователям предоставить доступ к данным той или иной категории. Например, самый высокий уровень безопасности может требоваться для документов и данных стратегического планирования предприятия, к которым должны иметь доступ только руководители предприятия и их помощники, далее могут идти важные финансовые и персональные данные, к которым может иметь доступ только администрация, и т. д.



Когда для данных определено несколько категорий или уровней безопасности, говорят о *многоуровневой системе безопасности.* Общим требованием многоуровневой системы безопасности является то, что субъект на более высоком уровне не может сообщить информацию субъекту на более низком или несравнимом уровне, если только соответствующий поток не вызван действием пользователя, имеющего право на доступ к данной информации.

Для упрощения реализации данное требование записывается в виде пары более простых требований, а именно предполагается, что в многоуровневой системе безопасности должны выполняться следующие требования:

- *Запрет чтения вверх. Любому субъекту разрешается выполнять операции чтения только в отношении объектов с тем же или более низким уровнем безопасности. Это требование часто называют простым требованием безо пасности (simple security property).*
- *Запрет записи вниз. Субъект может выполнять операции записи только в отношении объектов с тем же или более высоким уровнем безопасности. Это требование часто называют свойством * (* — property), что произносится как «свойство звездочка».*

Аккуратная реализациях обоих правил обеспечивает многоуровневую систему безопасности. Данный подход был апробирован в системах обработки данных, где после долгих исследований была выработана концепция *монитора обращений*

(reference monitor). Схема функционирования многоуровневой системы безопасности представлена на рис. 8.3.

Монитор обращений представляет собой контролирующий элемент, управляющий доступом субъектов к объектам на основе параметров безопасности субъектов и объектов, реализованный на аппаратном уровне и уровне операционной системы компьютера.

Монитор обращений имеет доступ к файлу, называемому *базой данных ядра безопасности,* где указаны привилегии доступа (категории допуска) субъектов и атрибуты защиты (грифы секретности) объектов.

Монитор обращений реализует два упомянутых выше требования (запрет чтения вверх и запрет записи вниз), и ему присущи следующие характеристики:

- *Неизбежное посредничество.* Правила безопасности навязываются при любой попытке получения доступа, а не только, скажем, при открытии файлов.
- *Изолированность.* Монитор обращений и его база данных защищены от внесения несанкционированных изменений.
- *Доказуемость.* Корректность работы монитора обращений должна быть доказуемой. Это значит, что должна иметься возможность доказать математически, что монитор обращений обеспечивает выполнение правил безопасности, гарантируя неизбежное посредничество и изолированность.

Эти требования являются жесткими. Требование неизбежной опосредованности означает, что любая операция доступа к данным в оперативной памяти, на диске или магнитной ленте должна быть опосредованной. Требование изолированности означает, что у нарушителя не должно быть возможности изменить логику работы монитора обращений или содержимое базы данных ядра безопасности. Система, для которой обеспечена математическая доказуемость, называется высоконадежной системой (trusted system).

В файле аудита регистрируются события, важные с точки зрения обеспечения безопасности, например выявленные нарушения безопасности или авторизованные изменения в базе данных ядра безопасности.

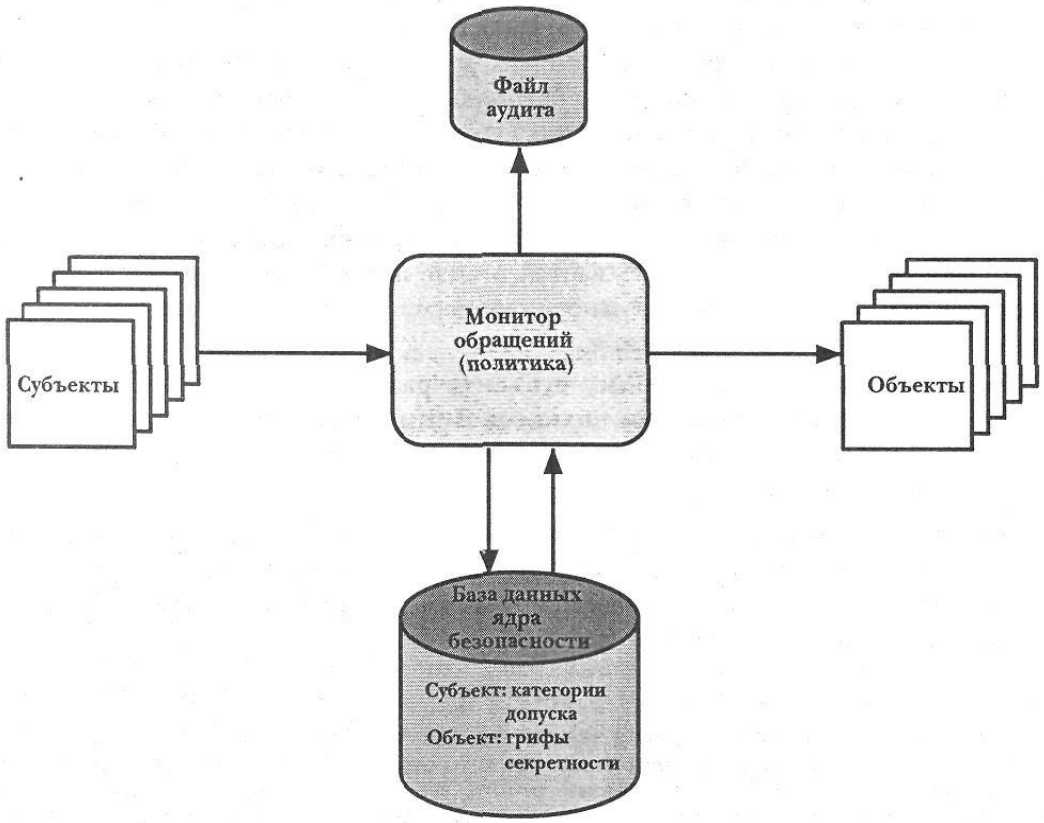


Рис. 8.3 – Схема функционирования многоуровневой системы безопасности

Контрольные вопросы по главе 8

1) Перечислите задачи, выполняемые брандмауэрами.
2) Приведите ограничения, существующие для брандмауэров.
3) Охарактеризуйте типы брандмауэров.
4) Приведите схему брандмауэра с экранированным одноточечным бастионным узлом.
5) Приведите схему брандмауэра с экранированным двухточечным бастионным узлом.
6) Приведите схему брандмауэра с экранированной подсетью.
7) Опишите фильтрующий маршрутизатор.
8) Опишите шлюз уровня приложений.
9) Опишите шлюз уровня коммутации.
10) Опишите бастионный узел.
11) Приведите конфигурации брандмауэров.

12) Опишите модель управления доступом к данным в высоконадежных системах.
13) Сформулируйте основные требования к обеспечению прав доступа в многоуровневых системах.
14) Приведите схему концепции монитора обращений.

ЗАКЛЮЧЕНИЕ

В пособии рассмотрены современные проблемы защиты информации и криптографические методы обеспечения безопасности вычислительной техники. Изложены математические основы криптографических алгоритмов и проведен анализ наиболее распространенных компьютерных алгоритмов защиты информации. Рассмотрены способы осуществления информационной безопасности отдельных рабочих станций и локальных компьютерных сетей.

Проведен анализ существующих типов вирусов и принципов построения систем антивирусной защиты в локальных и корпоративных сетях. Представлены существующие конфигурации брандмауэров, структура управления доступом к данным и принципы создания многоуровневых систем безопасности.

В результате освоения изложенного материала студент сможет самостоятельно приступить к обеспечению компьютерной безопасности своей рабочей станции и локальной информационно-вычислительной сети.

ЛИТЕРАТУРА

[1] Молдовян А. А. Криптография / А. А. Молдовян, Н. А. Молдовян, Б. Я. Советов. — СПб.: Лань, 2000. — 224 с.

[2] Петров А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Пертов. — М.: ДМК, 2000. — 448 с.

[3] Спицын В. Г. Защита информации и информационная безопасность / В. Г. Спицын, Н. А. Столярова. — Томск : Изд-во ТПУ. — 2003. — 167 с.

[4] Столлингс В. Криптография и защита сетей: принципы и практика / В. Столлингс. — М.: Издательский дом «Вильямс», 2001. — 672 с.

[5] Смарт Н. Криптография / Н. Смарт. — М.: Техносфера, 2005. — 528 с.

[6] Schneier B. Applied Cryptography: Protocols, Algorithms and Source Code in C / B. Schneier. — 2-е изд. — John Wiley& Sons, Inc, 1994. (Шнайер Брюс. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С / Брюс Швайнер. — 2-е изд. — 1994). — http://www.ssl.stu.neva.ru/psw/crypto.html

[7] Шеннон К. Э. Работы по теории информации и кибернетике / К. Э. Шеннон. — М.: Иностранная литература, 1963. — С. 333- 402.

[8] Виноградов И. М. Основы теории чисел / И. М. Виноградов. — М.: Наука, 1981. — 176 с.

[9] Кнут Д. Э. Искусство программирования для ЭВМ : в 3 т / Д. Э. Кнут. — М.: Мир, 2000. Т. 2. — 724 с.

ГЛОССАРИЙ

Асимметричный (двухключевой) шифр — включает криптографический алгоритм и два пространства возможных ключей: открытого и секретного.

ГОСТ 28147-89 — стандарт симметричного шифрования данных СССР и России, введенный в 1990 г. Алгоритм работает с 64-битными блоками. В нем применяется 256-битовый ключ и осуществляется 32 цикла шифрования.

Ключ — легкосменяемый элемент криптосистемы, являющийся секретным в симметричных шифрах и секретным или открытым в асимметричных (двухключевых) шифрах.

Криптографический алгоритм — долговременный элемент криптосистемы, обеспечивающий криптографическое преобразование данных.

Криптография — отрасль науки, изучающая математические методы создания криптографических алгоритмов обеспечения защиты информации от несанкционированного доступа.

Криптоанализ — отрасль науки, изучающая математические методы вскрытия криптографических алгоритмов и систем защиты информации.

Криптология — отрасль науки, изучающая математические методы создания и вскрытия криптографических алгоритмов. Криптология состоит из двух частей: криптографии и криптоанализа.

Криптосистема — включает в себя алгоритм криптографического преобразования и пространство возможных ключей.

Открытый текст — исходные данные (текст, звук, видео), готовые к применению без дополнительной обработки.

Симметричный шифр — включает криптографический алгоритм и пространство возможных секретных ключей.

Хэш-функция — осуществляет преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины.

Шифр — обратимое преобразование открытого текста в зашифрованный текст.

Шифрование — процесс применения криптографического преобразования открытого текста на основе алгоритма и ключа, в результате которого появляется зашифрованный текст.

Шифротекст — данные, полученные после криптографического преобразования открытого текста.

AES (RIJNDAEL) — новый стандарт симметричного шифрования данных США (2002 г.), основанный на алгоритме *RIJNDAEL*. Размер блока для шифрования составляет 128 бит. Применяются ключи шифрования трех фиксированных размеров 128, 192 и 256 бит. В *AES*, в зависимости от длины ключа, осуществляется 10, 12 или 14 циклов шифрования.

CBC (Cipher Block Chaining) — способ использования блочного шифра в режиме «сцепления блоков шифра». В этом случае происходит добавление к каждому блоку шифротекста контекстного идентификатора.

CFB (Cipher FeedBack) — режим, называемый «обратной связью по шифротексту». В нем блочный шифр трансформируется в поточный.

DES (Data Encryption Standard) — первый стандарт симметричного шифрования данных США, размер блока для шифрования составляет 64 бита, размер ключа — 56 бит. В нем осуществляется 16 циклов шифрования. Разработан фирмой IBM и утвержден правительством США как официальный стандарт в 1977 г.

ECB (Electronic Code Book) — простейший способ использования блочного шифра. В нем данные делятся на блоки и затем зашифровываются.

IDEA (I nternational D ata E ncryption A lgorithm) — симметричный алгоритм шифрования данных, размер блока для шифрования составляет 64 бита, размер ключа — 128 бит. В нем осуществляется 8 циклов шифрования.

MD5 — однонаправленная хэш-функция, получающая на вход строку произвольной длины. Результатом ее работы является 128-битовое хэш-значение. Создана в 1991 г. в Массачусетском технологическом институте США.

OFB (Output Feedback) — режим, называемый «обратной связью по выходу», адаптирует блочный шифр к его поточному использованию.

RSA — асимметричный (двухключевой) криптоалгоритм, применяемый как для зашифрования сообщений, так и для осуществления электронной цифровой подписи документов. Описание RSA было опубликовано в 1977 г. в журнале Scientific American.

PGP (Pretty Good Privacy) — комплекс криптоалгоритмов, первоначально предназначенный для ведения секретной переписки в Интернете. Разработан в 1991 г. Ф. Циммерманом.

Учебное издание

Спицын Владимир Григорьевич

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебное пособие

Корректор Осипова Е. А.
Компьютерная верстка Хомич С. Л.

Подписано в печать 07.11.11. Формат 60x84/8.
Усл. печ. л. 17,21. Тираж 300 экз. Заказ

Издано в ООО «Эль Контент»
634029, г. Томск, ул. Кузнецова д. 11 оф. 17
Отпечатано в Томском государственном университете
систем управления и радиоэлектроники.
634050, г. Томск, пр. Ленина, 40
Тел. (3822) 533018.