Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

П. П. Урбанович

ЗАЩИТА ИНФОРМАЦИИ МЕТОДАМИ КРИПТОГРАФИИ, СТЕГАНОГРАФИИ И ОБФУСКАЦИИ

Рекомендовано
учебно-методическим объединением по образованию
в области информатики и радиоэлектроники
в качестве учебно-методического пособия
для студентов учреждений высшего образования
по специальности 1-98 01 03 «Программное обеспечение
информационной безопасности мобильных систем»,
направлению специальности 1-40 05 01-03
«Информационные системы и технологии
(издательско-полиграфический комплекс)»,
специальности 1-40 01 01 «Программное обеспечение
информационных технологий» специализации
1-40 01 01 10 «Программирование Интернет-приложений»

Минск 2016

УДК [004.056+003.26](075.8)
ББК 32.97я73
У69

Р е ц е н з е н т ы :
кафедра электронных вычислительных средств
учреждения образования «Белорусский государственный университет
информатики и радиоэлектроники» (заведующий кафедрой
доктор технических наук, профессор *А. А. Петровский*);
директор учреждения «Главный информационно-аналитический центр
Министерства образования Республики Беларусь»
доктор технических наук, профессор *Н. И. Листопад*

Урбанович, П. П.

У69 Защита информации методами криптографии, стеганографии и обфускации : учеб.-метод. пособие для студентов специальности 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем», направления специальности 1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)», специальности 1-40 01 01 «Программное обеспечение информационных технологий» специализации 1-40 01 01 10 «Программмирование Интернет-приложений» / П. П. Урбанович. – Минск : БГТУ, 2016. – 220 с.

ISBN 978-985-530-562-1.

В издании изложены основные понятия, относящиеся к проблеме защиты информации в компьютерных системах и сетях, направления и методика разработки политики безопасности организаций и учреждений. Рассмотрены математические основы методов криптографии, стеганографии и обфускации и на конкретных примерах показаны алгоритмы и методики практической реализации этих методов для зашифрования/расшифрования данных, защиты права интеллектуальной собственности на текстовые документы, базы данных и коды программ, генерации электронной цифровой подписи и ее проверки.

Издание предназначено для студентов ИТ-специальностей, а также может быть полезно магистрантам, научным работникам, объектами исследований которых являются методы и средства обеспечения информационной безопасности.

УДК [004.056+003.26](075.8)
ББК 32.97я73

ISBN 978-985-530-562-1

ПРЕДИСЛОВИЕ

За последние 10–15 лет информационные технологии существенно расширили и усилили свой «плацдарм» во всех сферах нашей жизнедеятельности. Это обстоятельство – безусловный положительный фактор, влияющий на инновационный характер развития реального сектора экономики, здравоохранения, сферы услуг, досуга и, конечно же, образования. Однако наряду с этим указанный тренд со все большей очевидностью обнажает остроту проблем, негативных последствий информатизации. В наибольшей степени эти проблемы связаны с возможностями несанкционированного доступа к информационным ресурсам, объектам инфраструктуры, принадлежащим другим физическим лицам, субъектам хозяйствования, банковской сферы, другим государствам. Это напрямую связано с необходимостью обеспечения не только информационной, но и государственной безопасности.

Указанные причины возлагают на систему ИТ-образования ответственность в части не только информатизации самой сферы образования, но и подготовки специалистов, способных эффективно решать указанные проблемы.

В рамках существующих стандартов образования и учебных планов ряда специальностей (в том числе направления специальности 1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)», 1-40 01 01 «Программное обеспечение информационных технологий» (специализация 1-40 01 01 10 «Программирование Интернет-приложений»), 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем») предусматривается изучение студентами комплекса специальных дисциплин, направленных на овладение теоретическими знаниями и практическими навыками в анализируемой предметной области. К числу таких дисциплин относятся «Защита информации и надежность информационных систем» (направление специальности 1-40 05 01-03), «Криптографические методы защиты информации» (специальность 1-98 01 03, специализация 1-40 01 01 10).

С учетом того обстоятельства, что динамика процессов в ИТ-сфере требует адекватного реагирования путем внесения соответствующих изменений в учебные планы, подготовки учебно-

методических пособий, следует признать, что методическое обеспечение учебных дисциплин, относящихся к информационной безопасности, в учреждениях высшего образования пока не носит системного характера. Данное учебно-методическое пособие, по мнению автора, призвано восполнить указанный пробел.

Содержание книги охватывает вопросы, относящиеся к трем основным классам методов преобразования информации с целью ее защиты от несанкционированного доступа и использования, нарушения целостности, а также защиты права интеллектуальной собственности: криптографии, стеганографии и обфускации.

Материал издания излагается в последовательности, которая, по мнению автора, является оптимальной с методической точки зрения при изучении его в полном объеме, хотя данная точка зрения не претендует на бесспорность. С другой стороны, каждый раздел оформлен в определенной степени автономно, что может повысить эффективность изучения избранных разделов.

После глав, а в некоторых случаях и после подглав даны вопросы для самостоятельного контроля знаний либо контроля знаний студента преподавателем. На основе сформулированных вопросов и заданий могут быть составлены тесты для компьютерного (дистанционного) контроля знаний. Наш опыт показывает, что такая форма контроля достаточно эффективна не только для студентов, обучающихся по заочной (или дистанционной) форме образования, но и студентов очной формы.

Раздел 3 пособия написан автором совместно с Н. П. Шутько, раздел 5 – совместно с В. А. Пласковицким.

Раздел I

МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ И ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ СИСТЕМ

Глава 1

ФУНДАМЕНТАЛЬНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ИЗ ОБЛАСТИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ СИСТЕМ

1.1. Краткая историческая справка

Развитие информационных технологий и оценку их безопасности обычно связывают со следующими этапами.

I этап – примерно до середины XV века, когда Иоганн Гутенберг (Johannes Gutenberg) создал европейский способ книгопечатания подвижными литерами, распространившийся по всему миру. До этого времени практически любая информация передавалась устно, т. е. конфиденциальность определялась исключительно «человеческим фактором».

II этап – примерно до начала XIX века – характеризуется использованием естественно возникавших средств информационных коммуникаций. Основная задача информационной безопасности – защита сведений о событиях, фактах, имуществе и т. д.

III этап – связан с началом использования технических средств электро- и радиосвязи. Характеризуется применением помехоустойчивого кодирования сообщения (сигнала) с последующим декодированием принятого сообщения (сигнала).

IV этап – начиная с 1935 года – связан с появлением радиолокационных и гидроакустических средств. Обеспечение информационной безопасности основывалось на сочетании организационных и технических мер, направленных на повышение защищенности радиолокационных средств от воздействия на их приемные устройства активных и пассивных помех.

V этап – начиная с 1946 года – связан с изобретением и внедрением в практическую деятельность электронно-вычислительных машин (компьютеров). Эру появления компьютерной техники связывают с разработкой в Пенсильванском университете (США) ЭВМ ENIAC (*Electronic Numerical Integrator And Computer (Calculator)*). Задачи информационной безопасности решались в основном методами и способами ограничения физического доступа к оборудованию средств сбора, переработки и передачи информации.

VI этап – начиная с первой половины 60-х годов XX века – обусловлен созданием и развитием локальных информационно-коммуникационных сетей. Задачи безопасности решались в основном методами и способами физической защиты средств, путем администрирования и управления доступом к сетевым ресурсам.

VII этап – начиная с 1973 года – связан с использованием мобильных коммуникационных устройств с широким спектром задач. В этот период созданы известные сейчас во всем мире фирмы Microsoft (Билл Гэйтс и Пол Ален) и Apple (Стив Джобс и Стэфан Возняк).

Образовались сообщества людей – *хакеров*, ставящих своей целью нанесение ущерба информационной безопасности отдельных пользователей, организаций и целых стран. Формируется *информационное право* – новая отрасль международной правовой системы.

VIII этап – начиная примерно с 1985 года – связан с созданием и развитием глобальных информационно-коммуникационных сетей с использованием космических средств обеспечения; предусматривает комплексное использование мер и средств защиты.

IX этап – примерно с конца XX – начала XXI века – связан с повсеместным использованием сверхмобильных коммуникационных устройств с широким спектром задач и глобальным охватом в пространстве и времени, обеспечиваемым космическими информационно-коммуникационными системами. Характеризуется *«широким переходом на цифру»*, применением *облачных технологий*, хранением, обработкой и использованием больших объемов данных (*Big Data*); предусматривает комплексное использование мер и средств защиты.

Проблема защиты информации путем ее преобразования относится к криптологии (греч. κρυπτός (*kryptos*) – тайный, λόγος (*logos*) – слово). Криптология разделяется на два направления – криптографию и криптоанализ. Цели этих направлений прямо противоположны.

Криптография (греч. κρυπτός и γράφω (*grafo*) – пишу), или тайнопись, занимается поиском и исследованием математических

методов преобразования информации. Сфера интересов *криптоанализа* – исследование возможности расшифровывания информации или взлома шифров.

Криптография – одна из старейших наук, ее история насчитывает несколько тысяч лет. Первоначально письменность сама по себе была криптографической системой, так как в древних обществах ею владели только избранные. Священные книги Древнего Египта, Древней Индии тому примеры. С широким распространением письменности криптография стала формироваться как самостоятельная наука. Первые криптосистемы встречаются уже в начале нашей эры. Бурное развитие криптографические системы получили в годы Первой и Второй мировых войн.

В конце 1918 года в свет вышел один из самых значительных трудов XX столетия в области криптоанализа – монография Уильяма Ф. Фридмана (William F. Friedman) «Индекс совпадения и его применение к криптографии» (*Index of Coincidense and its Application in Cryptograph*y)[1]. Примерно в то же время Эллард Х. Хеберн (США) получил первый патент (US Patent № 1510441) на роторную машину – устройство, которому суждено было стать основой военной криптографии почти на полвека.

На протяжении 1930–1940-х годов работы в данной области в открытой печати практически не публиковались. В это время Клод Шеннон (Claude Shannon), которого принято считать отцом теории информации, опубликовал статью «Теория связи в секретных системах»[2] (*The Communication Theory of Secrecy Systems*), которая послужила началом обширных исследований в теории кодирования и передачи информации и, по всеобщему мнению, придала криптографии статус науки.

В конце 1960-х – начале 1970-х годов фирма IBM опубликовала ряд отчетов своих сотрудников, в частности Х. Файстеля (H. Feisttel), Дж. Смита (J. Smith) и др. На основе этих исследований впоследствии был разработан стандарт симметричного шифрования данных DES (Data Encryption Standard).

В 1975 году У. Диффи (W. Diffie) и М. Хеллман (M. Hellman) предложили новый метод криптографического преобразования

---

[1] Переиздано: Friedman W. F. The Index of Coincidence and Its Application in Cryptography. Laguna Hills, CA: Aegean Park Press, 1986. 101 p.

[2] Shannon C. Communication Theory of Secrecy Systems // Bell System Technical Journal. 1949. Vol. 28 (4). P. 656–715.

информации – криптографию с открытым ключом, или асимметричную криптографию[3].

В последние десятилетия основным объектом исследований криптографической науки стала не теория, а практика. Это обусловлено несколькими причинами, основной среди них является развитие Интернет-технологий.

1.2. Основные понятия и определения из области защиты информации

Далее сформулируем определения основных понятий, которыми будем пользоваться при изучении предметной области.

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

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

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

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

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

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

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

Несанкционированный доступ (НСД) – доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи,

[3] Diffie W., Hellman M. E. New Directions in Cryptography // IEEE Transactions on Information Theory. 1976. Vol. IT-22. P. 644–654.

реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные правила доступа.

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

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

Атака – попытка несанкционированного преодоления защиты системы.

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

Другое определение:

информационная безопасность – все аспекты, связанные с определением, достижением и поддержанием конфиденциальности, целостности, доступности информации или средств ее обработки:

1) *конфиденциальность (confidentiality)* – состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на нее право;

2) *целостность (integrity)* – избежание несанкционированной модификации информации;

3) *доступность (availability)* – избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.

Идентификация – процесс распознавания определенных компонентов системы (объектов или субъектов) с помощью уникальных идентификаторов.

Аутентификация – проверка идентификации пользователя или иного компонента ИС для принятия решения о разрешении доступа к ресурсам системы.

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

Ошибка устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или

воздействующими на него помехами (преднамеренными либо непреднамеренными), или иным способом.

Ошибка программы – проявляется в не соответствующем реальному (требуемому) промежуточном или конечном значении (результате) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

1.3. Общая характеристика факторов, влияющих на безопасность и надежность ИВС

Фактор, *воздействующий на ИВС*, – это явление, действие или процесс, результатом которых может быть утечка, искажение, уничтожение данных, блокировка доступа к ним, повреждение или уничтожение системы защиты.

Все многообразие дестабилизирующих факторов можно разделить на два класса: внутренние и внешние.

Внутренние дестабилизирующие факторы влияют:

1) на программные средства (ПС):

а) некорректный исходный алгоритм;

б) неправильно запрограммированный исходный алгоритм (первичные ошибки);

2) на аппаратные средства (АС):

а) системные ошибки при постановке задачи проектирования;

б) отклонения от технологии изготовления комплектующих изделий и АС в целом;

в) нарушение режима эксплуатации, вызванное внутренним состоянием АС.

Внешние дестабилизирующие факторы влияют:

1) на программные средства:

а) неквалифицированные пользователи;

б) несанкционированный доступ к ПС с целью модификации кода;

2) на аппаратные средства:

а) внешние климатические условия;

б) электромагнитные и ионизирующие помехи;

в) перебои в электроснабжении;

г) недостаточная квалификация обслуживающего персонала;

д) несанкционированный (в том числе – удаленный) доступ с целью нарушения работоспособности АС.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Дайте определения основных понятий и терминов, относящихся к области защиты информации и надежности информационных систем.
2. Чем отличается идентификация от авторизации?
3. Охарактеризуйте основные этапы развития информационных технологий с точки зрения их безопасности.
4. Приведите классификацию основных факторов, влияющих на ИВС.
5. К каким последствиям приводит влияние дестабилизирующих факторов на ИС (ИВС)?
6. Дайте характеристику внутренних факторов, дестабилизирующих работу ИС (ИВС).
7. Охарактеризуйте внешние факторы, дестабилизирующие работу ИС (ИВС).
8. Как Вы понимаете «некорректный исходный алгоритм»?
9. Что такое «первичная ошибка» в программе?
10. Как влияют климатические условия на надежность аппаратных средств?

Глава 2

ПОТЕНЦИАЛЬНЫЕ УГРОЗЫ БЕЗОПАСНОСТИ ИНФОРМАЦИИ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ. ОБЪЕКТЫ И МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

2.1. Естественные и искусственные помехи и угрозы безопасности

Одним из важнейших дестабилизирующих работу ИВС факторов считают *электромагнитные* и *ионизирующие* излучения. Источниками первых являются практически все средства, функ-

ционирование которых основано на использовании электроэнергии, и в особенности такие АС, которые целенаправленно излучают электромагнитные волны (к ним относится, например, приемо-передающие и иные подобные радиоэлектронные устройства). По большому счету любой проводник с током является источником электромагнитных помех. Такие источники относятся к классу *искусственных* или *промышленно-бытовых*. В свою очередь, их можно подразделить на *непреднамеренные* и *преднамеренные*. Последние имеют место в ситуациях, похожих на эпизод в фильме о приключениях Шурика: профессор на экзамене включил генератор помех, который «забил» канал.

Ионизирующие излучения также могут иметь естественную (солнечная радиация) и искусственную (изотопы урана и тория излучают даже пластмассы) природу.

Основным последствием влияния помех на АС являются ошибки в хранящейся, передаваемой или обрабатываемой информации. Другими словами – **помехи снижают функциональную надежность АС**.

Для лучшего понимания сути и особенностей *угроз со стороны деструктивных* или *злонамеренных программных средств* (англ. *malware* – ***mal****icious soft****ware***), а также физических лиц, использующих такие средства для организации НСД (хакеры и кракеры), на рис. 2.1 приведен пример периметра современной ИС, а на рис. 2.2 схематически показаны наиболее уязвимые места локальной сети.

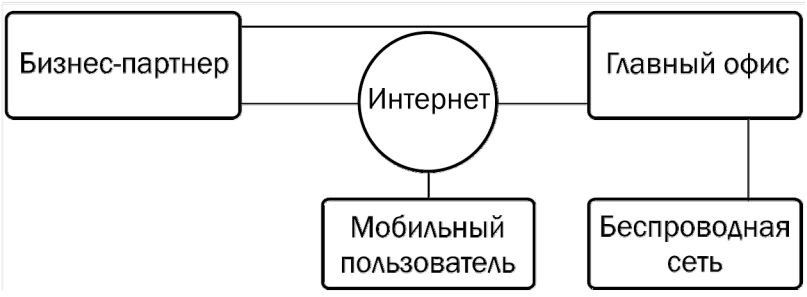


Рис. 2.1. Пример периметра современной ИС

Основные факторы (угрозы):

1) действия злоумышленника;
2) наблюдение за источниками информации;
3) подслушивание конфиденциальных разговоров и акустических сигналов работающих механизмов;

4) перехват электрических, магнитных и электромагнитных полей, электрических сигналов и радиоактивных излучений;

5) несанкционированное распространение материальных носителей за пределами организации;

6) разглашение информации компетентными людьми;

7) утеря носителей информации;

8) несанкционированное распространение информации через поля и электрические сигналы, случайно возникшие в аппаратуре;

9) воздействие стихийных сил (наводнения, пожары и т. п.);

10) сбои и отказы в аппаратуре сбора, обработки и передачи информации;

11) отказы системы электроснабжения;

12) воздействие мощных электромагнитных и электрических помех (промышленных и природных).

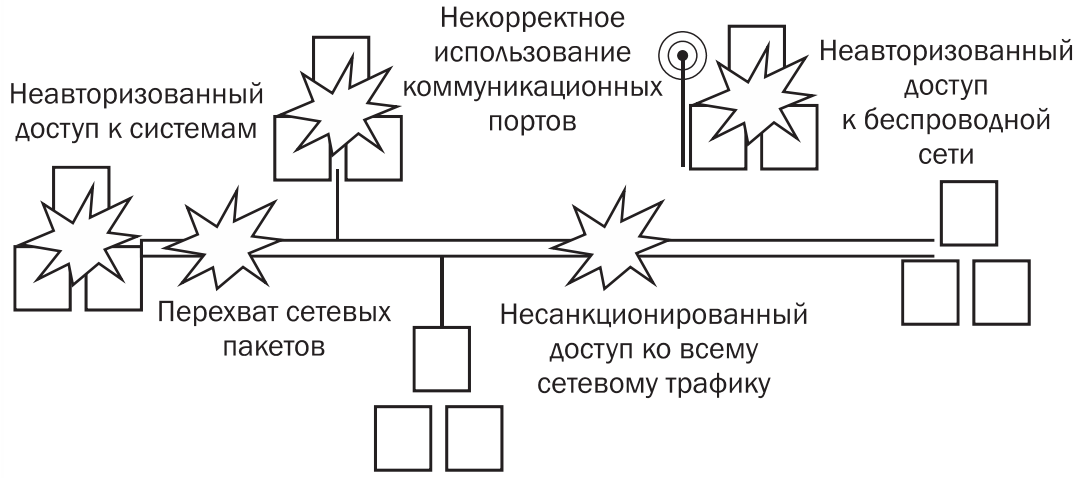


Рис. 2.2. Наиболее уязвимые места локальной сети

Несанкционированный доступ с помощью *деструктивных программных средств* осуществляется, как правило, через компьютерные сети.

Классификацию вредоносного ПО можно представить следующим образом:

вирусы (viruses) – это **саморазмножающиеся программы** путем дописывания собственных кодов к исполняемым файлам; вирусы могут содержать деструктивные функции;

черви (worms) – это программы, которые самостоятельно размножаются по сети и, в отличие от вирусов, не дописывают себя (как правило) к исполняемым файлам; все черви «съедают» ресур-

сы компьютера, «нагоняют» интернет-трафик и могут привести к утечке данных с вашего компьютера;

анализаторы клавиатуры, или *кейлоггеры* (keyloggers), – программы, которые регистрируют нажатия клавиш, делают снимки рабочего стола, отслеживают действия пользователя во время работы за компьютером и сохраняют эти данные в скрытый файл на диске, затем этот файл попадает к злоумышленнику;

трояны (trojans), или *троянские кони*, – собирают конфиденциальную информацию с компьютера пользователя (пароли, базы данных и пр.) и тайно по сети высылают их злоумышленнику (своему хозяину);

боты (bots) – распространенный в наше время вид зловредного ПО, который устанавливается на компьютерах пользователей (*сети botnet*) и используется для атак на другие компьютеры;

снифферы (sniffers) – это анализаторы сетевого трафика; могут использоваться в составе зловредного ПО, скрытно устанавливаться на компьютере пользователя и отслеживать данные, которые отправляет или получает пользователь по сети;

руткиты (rootkits) – сами по себе не являются зловредным ПО; назначение – скрывать работу других зловредных программ (кейлоггеров, троянов, червей и т. д.) как от пользователя, так и от программ и средств обеспечения безопасности (антивирусов, файерволов (firewalls), систем обнаружения атак и пр.).

Важно отметить, что вирусы, трояны и иные деструктивные программы активизируются только после загрузки инфицированного файла в оперативную память компьютера (RAM).

Принято считать, что основную угрозу таят 4 группы вредоносных программ: вирусы (или классические вирусы), сетевые черви, троянские кони и хакерские программы.

Вирусы начинают свою «подрывную» работу после выполнения пользователем определенных действий, например при неосмотрительном запуске неизвестного файла. Много проблем могут доставить полиморфные вирусы: их программный код постоянно меняется, что делает их обнаружение классическими способами практически невозможным. Обезвредить такие вирусы можно только с использованием методов, действие которых основывается на анализе общих для всех вирусов характеристик поведения. Сетевые черви «расползаются» по локальным или глобальным сетям и после удачного проникновения на ПК действуют самостоятель-

но, так как являются автономными программами. В категорию троянских коней входят программы для хищения информации. Хакерские утилиты проникают на компьютер с заранее запрограммированными злоумышленниками задачами. Одна из их основных целей – взлом защиты и предоставление хакеру-хозяину удаленного доступа к ресурсам жертвы.

Все чаще пользователи сталкиваются с *кибермошенничеством* и другими видами сетевых угроз, подключаясь к Интернету через бесплатные точки доступа Wi-Fi, например, для проверки банковского счета, оплаты мобильного телефона, отправки почты и других действий.

Выделяют несколько основных угроз безопасности, возникающих при использовании бесплатных точек доступа Wi-Fi:

– сети, организованные хакерами, могут выдавать себя за вполне легальные бесплатные точки доступа;

– атаки с помощью вредоносного ПО компьютера, подключенного к этой точке доступа;

– *сниффинг* (сниффер (от англ. *to sniff* – нюхать) – сетевой анализатор трафика, программа или программно-аппаратное устройство, предназначенное для перехвата и последующего анализа либо только анализа сетевого трафика, предназначенного для других узлов), целью является перехват и анализ злоумышленниками Интернет-трафика пользователя, что приводит к утере конфиденциальных данных;

– хищение персональной информации методом «*человек посередине*» (*man in the middle*), что означает ситуацию, при которой злоумышленник может читать сообщения, которыми обмениваются пользователи, так чтобы они даже не догадывалась о его присутствии.

В связи с этим не рекомендуется использовать неизвестные точки доступа Wi-Fi, а также по возможности следует минимизировать количество конфиденциальной информации, хранящейся на мобильном устройстве. И, что особенно важно, для совершения банковских операций необходимо использовать только проверенные точки доступа к сети Интернет. Для доступа к электронной почте подключение к сети Wi-Fi следует осуществлять через защищенный протокол https, а не http.

Особым видом мошенничества является фишинг.

Фишинг (англ. *phishing*, от *fishing* – рыбная ловля) – вид Интернет-мошенничества, целью которого является получение дос-

тупа к конфиденциальным данным пользователей – логинам и паролям. Это достигается путем проведения массовых рассылок электронных писем от имени популярных брендов, а также личных сообщений внутри различных сервисов, например от имени банков или внутри социальных сетей. В письме часто содержится прямая ссылка на сайт, внешне неотличимый от настоящего. После того как пользователь попадает на поддельную страницу, мошенники пытаются различными психологическими приемами побудить пользователя ввести на поддельной странице свои логин и пароль, которые он использует для доступа к определенному сайту, что позволяет мошенникам получить доступ, например, к банковским счетам.

По данным международной компании «Лаборатория Касперского», к числу стран, наиболее часто подвергавшихся в 2014 году атакам банковскими троянами, относятся в основном страны бывшего СССР. По тем же аналитическим данным карта попыток заражений мобильными зловредными программами, полученная по 213 странам мира, использующим антивирусное ПО «Лаборатории Касперского», выглядит так, как представлено в табл. 2.1 (процент от всех атакованных уникальных пользователей).

Таблица 2.1

Карта выявленных попыток заражений ИС мобильными зловредными программами

Страна	Процент атак
Россия	48,9
Индия	5,23
Казахстан	4,55
Украина	3,27
Великобритания	2,79
Германия	2,70
Вьетнам	2,44
Малайзия	1,79
Испания	1,58
Польша	1,54

Наиболее уязвимыми с точки зрения защищенности ресурсов являются так называемые критические информационные системы.

Критическая информационная сиситема (КИС) – это сложная компьютеризированная организационно-техническая систе-

ма, блокировка или нарушение функционирования которой потенциально приводит к потере устойчивости организационных систем государственного управления и контроля, утрате обороноспособности государства, разрушению системы финансового обращения, дезорганизации систем энергетического и коммуникационно-транспортного обеспечения государства, глобальным экологическим или техногенным катастрофам.

При решении проблемы повышения уровня защищенности информационных ресурсов КИС необходимо исходить из того, что наиболее вероятным информационным объектом воздействия будет выступать программное обеспечение, составляющее основу комплекса средств получения, семантической переработки, распределения и хранения данных, используемых при эксплуатации таких систем.

При рассмотрении правовых аспектов защиты информации и информационной безопасности противоправные действия классифицируются как *компьютерное преступление*. Наказания за преступления против информационной безопасности впервые были юридически закреплены в Уголовном кодексе Республики Беларусь в 1999 году.

В *уголовном праве Беларуси* закреплена ответственность за ряд преступлений против информационной безопасности (глава 31 Уголовного кодекса Республики Беларусь). К преступлениям, связанным с использованием компьютерных средств (подлог и мошенничество, совершенные с использованием компьютерных технологий) в Уголовном кодексе Республики Беларусь относят хищение путем использования компьютерной техники (ст. 212); причинение имущественного ущерба без признаков хищения путем модификации компьютерной информации (ст. 216); ряд составов преступления, которые включают хищение (ст. 294, 323, 327, 333). Третья группа преступлений включает преступления, связанные с содержанием компьютерных данных: детская порнография, нарушение авторского права.

В отношении нарушений *авторских прав* с использованием компьютерных технологий уголовное право Республики Беларусь не предусматривает каких-либо специальных норм. Некоторые из рассмотренных составов преступлений имеют свои аналоги в *Кодексе об административных правонарушениях* (ст. 22.6 КоАП Республики Беларусь – несанкционированный доступ к компьютер-

ной информации; ст. 10.7 КоАП Республики Беларусь – причинение имущественного ущерба)[4].

В целом перечень преступлений против информационной безопасности, зафиксированный в законодательных и нормативных актах Республики Беларусь, соответствует положениям Будапештской конвенции (2001 г.)[5]. Конвенция охватывает широкий круг вопросов, в том числе все аспекты *киберпреступности*, включая незаконный доступ к компьютерным системам и перехват данных, воздействие на данные, воздействие на работу системы, противозаконное использование устройств, подлог и мошенничество с использованием компьютерных технологий, правонарушения, связанные с детской порнографией, и правонарушения, связанные с авторским правом и смежными правами. При подготовке конвенции преследовались цели формирования общей правоохранительной системы для борьбы с киберпреступностью и создания условий для обмена информацией между всеми странами, подписавшими конвенцию. В Будапештской конвенции также устанавливаются требования защиты прав и свободы каждого в сети Интернет.

Международная организация уголовной полиции (Интерпол) пользуется *классификацией компьютерных преступлений* по кодификатору международной уголовной полиции генерального секретариата Интерпола. В 1991 году данный кодификатор был интегрирован в автоматизированную систему поиска и в настоящее время доступен подразделениям Национальных центральных бюро Интерпола более чем 120 стран мира.

Все коды, характеризующие компьютерные преступления, имеют идентификатор, начинающийся с буквы ***Q***. Для характеристики преступления могут использоваться до пяти кодов, расположенных в порядке убывания значимости совершенного преступления:

1) QA – несанкционированный доступ и перехват:
   QAH – компьютерный абордаж,
   QAI – перехват,

---

[4] Информационная безопасность: как предотвратить преступления // Блог о свободном интернете [Электронный ресурс]. 2014. URL: http://www.law-trend.org/information-access/blog-information-access/informatsionnaya-bezopasnost-prestupleniya-i-nakazaniya (дата обращения: 10.02.2015).

[5] Конвенция о компьютерных преступлениях // Официальный сайт Совета Европы [Электронный ресурс]. 2001. URL: http://conventions.coe.int/treaty/rus/Treaties/Html/185.htm (дата обращения: 10.02.2015).

QAT – кража времени,
QAZ – прочие виды несанкционированного доступа и перехвата;
2) QD – использование деструктивных программных средств:
QDL – логическая бомба,
QDT – троянский конь,
QDV – компьютерный вирус,
QDW – компьютерный червь,
QDZ – прочие виды;
3) QF – компьютерное мошенничество:
QFC – мошенничество с банкоматами,
QFF – компьютерная подделка,
QFG – мошенничество с игровыми автоматами,
QFM – манипуляции с программами ввода-вывода,
QFP – мошенничество с платежными средствами,
QFT – телефонное мошенничество,
QFZ – прочие компьютерные мошенничества;
4) QR – незаконное копирование (пиратство):
QRG – компьютерные игры,
QRS – прочее программное обеспечение,
QRT – топография полупроводниковых изделий,
QRZ – прочее незаконное копирование;
5) QS – компьютерный саботаж:
QSH – с аппаратным обеспечением,
QSS – с программным обеспечением,
QSZ – прочие виды саботажа;
6) QZ – прочие компьютерные преступления:
QZB – с использованием компьютерных досок объявлений,
QZE – хищение информации, составляющей коммерческую тайну,
QZS – передача информации конфиденциального характера,
QZZ – прочие компьютерные преступления.

2.2. Основные методы и средства повышения безопасности ИС и ИВС

В контексте сформулированной цели здесь кратко проанализируем методы и средства повышения информационной безопасности (ИБ) систем.

Политика информационной безопасности систем, как и во всех подобных случаях, должна строиться на основе *системного подхода*, предусматривающего всесторонний анализ причин и угроз безопасности, оценки их последствий, необходимости, экономической или иной целесообразности и адекватности принимаемых противодействий.

Все многообразие используемых методов и средств защиты можно разделить на три класса:

– законодательная и нормативно-правовая база;

– организационно-технические и режимные меры и методы (политика информационной безопасности);

– аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ[6].

Законодательная и нормативно-правовая база.

1. Акты национального законодательства:

а) международные договоры Республики Беларусь;

б) Конституция Республики Беларусь;

в) законы Республики Беларусь, например *Закон Республики Беларусь от 10 ноября 2008 г. № 455-З* «Об информации, информатизации и защите информации»;

г) указы Президента Республики Беларусь, например *Указ Президента Республики Беларусь от 30 сентября 2010 г. № 515* «О некоторых мерах по развитию сети передачи данных в Республике Беларусь», *Указ Президента Республики Беларусь от 1 февраля 2010 г. № 60* «О мерах по совершенствованию использования национального сегмента сети Интернет», *указ Президента Республики Беларусь от 23 января 2014 г. № 46* «Об использовании государственными органами и иными государственными организациями телекоммуникационных технологий».

Концептуальные аспекты информационной безопасности в нашей стране приведены в *Указе Президента Республики Беларусь от 9 ноября 2010 г. № 575* «Об утверждении Концепции национальной безопасности Республики Беларусь», где, в частности, дана характеристика внутренним источникам угроз национальной безопасности, к числу которых относятся:

---

[6] Иногда это класс разделяют на два: технические методы и средства; программно-аппаратные средства.

• распространение недостоверной или умышленно искаженной информации, способной причинить ущерб национальным интересам Республики Беларусь;

• зависимость Республики Беларусь от импорта информационных технологий, средств информатизации и защиты информации, неконтролируемое их использование в системах, отказ или разрушение которых может причинить ущерб национальной безопасности;

• несоответствие качества национального контента мировому уровню;

• недостаточное развитие государственной системы регулирования процесса внедрения и использования информационных технологий;

• рост преступности с использованием информационно-коммуникационных технологий;

• недостаточная эффективность информационного обеспечения государственной политики;

• несовершенство системы обеспечения безопасности критически важных объектов информатизации.

Некоторые внешние источники угроз национальной безопасности Республики Беларусь:

• открытость и уязвимость информационного пространства Республики Беларусь от внешнего воздействия;

• доминирование ведущих зарубежных государств в мировом информационном пространстве, монополизация ключевых сегментов информационных рынков зарубежными информационными структурами;

• информационная деятельность зарубежных государств, международных и иных организаций, отдельных лиц, наносящая ущерб национальным интересам Республики Беларусь, целенаправленное формирование информационных поводов для ее дискредитации;

• нарастание информационного противоборства между ведущими мировыми центрами силы, подготовка и ведение зарубежными государствами борьбы в информационном пространстве;

• развитие технологий манипулирования информацией;

• препятствование распространению национального контента Республики Беларусь за рубежом;

• широкое распространение в мировом информационном пространстве образцов массовой культуры, противоречащих общечеловеческим и национальным духовно-нравственным ценностям;

• попытки несанкционированного доступа извне к информационным ресурсам Республики Беларусь, приводящие к причинению ущерба ее национальным интересам;

д) постановления Правительства Республики Беларусь, например *постановление Совета Министров Республики Беларусь от 11 февраля 2006 г. № 192* «Об утверждении Положения о сопровождении Интернет-сайтов республиканских органов государственного управления, иных государственных организаций, подчиненных Правительству Республики Беларусь»; *постановление Совета Министров Республики Беларусь от 11 августа 2011 г. № 1084* «О внесении изменений и дополнений в постановление Совета Министров Республики Беларусь от 29 апреля 2010 г. № 644»; *постановление Совета Министров Республики Беларусь от 26 мая 2009 г. № 675* «О некоторых вопросах защиты информации»; *постановление Совета Министров Республики Беларусь от 26 мая 2009 г. № 673* «О некоторых мерах по реализации Закона Республики Беларусь “Об информации, информатизации и защите информации” и о признании утратившими силу некоторых постановлений Совета Министров Республики Беларусь»;

е) нормативные правовые акты министерств и ведомств, например *постановление Оперативно-аналитического центра при Президенте Республики Беларусь и Министерства связи и информатизации Республики Беларусь от 29 июня 2010 г. № 4/11* «Об утверждении положения о порядке ограничения доступа пользователей Интернет-услуг к информации, запрещенной к распространению в соответствии с законодательными актами»; *приказ Оперативно-аналитического центра при Президенте Республики Беларусь от 2 августа 2010 г. № 60* «Об утверждении положения о порядке определения поставщиков Интернет-услуг, уполномоченных оказывать Интернет-услуги государственным органам и организациям, использующим в своей деятельности сведения, составляющие государственные секреты»; *приказ Оперативно-аналитического Центра при Президенте Республики Беларусь от 27 мая 2013 г. № 33* «Об утверждении Инструкции о порядке взаимодействия ведомственных систем электронного документооборота с системой межведомственного электронного документооборота государственных органов»; *приказ Оперативно-аналитического Центра при Президенте Республики Беларусь от 30 августа 2013 г. № 62* «О некоторых вопросах технической и криптографической

защиты информации»; *приказ Оперативно-аналитического Центра при Президенте Республики Беларусь от 10 декабря 2015 г. № 11* «Об утверждении Положения о Государственной системе управления открытыми ключами проверки электронной цифровой подписи Республики Беларусь»;

ж) нормативные правовые акты субъектов, органов местного самоуправления и т. д.

Правовой элемент системы организации защиты информации на предприятии основывается на нормах информационного права и предполагает юридическое закрепление взаимоотношений фирмы и государства по поводу правомерности использования системы защиты информации, фирмы и персонала по поводу обязанности персонала соблюдать установленные меры защитного характера, ответственности персонала за нарушение порядка защиты информации.

Правовая защита включает:

• наличие в организационных документах фирмы, правилах внутреннего трудового распорядка, трудовых договорах, в должностных инструкциях положений и обязательств по защите конфиденциальной информации;

• формулирование и доведение до сведения всех сотрудников положения о правовой ответственности за разглашение конфиденциальной информации, несанкционированное уничтожение или фальсификацию документов;

• разъяснение лицам, принимаемым на работу, положения о добровольности принимаемых ими на себя ограничений, связанных с выполнением обязанностей по защите информации.

Основные подсистемы защиты информации в правовом плане:

• установление на объекте режима конфиденциальности;

• разграничение доступа к информации;

• правовое обеспечение процесса защиты информации;

• четкое выделение конфиденциальной информации как основного объекта защиты.

Собственные *нормативно-правовые документы предприятия*, ориентированные на обеспечение информационной безопасности:

• политика информационной безопасности;

• положение о коммерческой тайне;

• положение о защите персональных данных;

• перечень сведений, составляющих конфиденциальную информацию;

• инструкция о порядке допуска сотрудников к сведениям, составляющим конфиденциальную информацию;

• положение о специальном делопроизводстве и документообороте;

• обязательство сотрудника о сохранении конфиденциальной информации;

• памятка сотруднику о сохранении коммерческой тайны.

2. Международные стандарты, например:

а) BS 7799-1:2005 – Британский стандарт BS 7799 Part 1 – *Code of Practice for Information Security Management* («Практические правила управления информационной безопасностью») – описывает 127 механизмов контроля, необходимых для построения системы управления информационной безопасностью организации, определенных на основе лучших примеров мирового опыта в данной области. Этот документ служит практическим руководством по созданию системы управления информационной безопасностью (СУИБ);

б) BS 7799-2:2005 – Британский стандарт BS 7799 Part 2 – *Information Security Management – Specification for Information Security Management Systems* («Спецификация системы управления информационной безопасностью») – определяет спецификацию СУИБ. Вторая часть стандарта используется в качестве критериев при проведении официальной процедуры сертификации СУИБ организации;

в) ISO/IEC 17799:2005 – «Информационные технологии – Технологии безопасности – Практические правила менеджмента информационной безопасности». Международный стандарт, базирующийся на BS 7799-1:2005;

г) ISO/IEC 27001:2005 – «Информационные технологии – Методы обеспечения безопасности – Системы управления информационной безопасностью – Требования». Международный стандарт, базирующийся на BS 7799-2:2005;

д) ISO/IEC 27002, сейчас: ISO/IEC 17799:2005 – «Информационные технологии – Технологии безопасности – Практические правила менеджмента информационной безопасности». Дата выхода – 2007 год;

е) ISO/IEC 27005, сейчас: BS 7799-3:2006 – руководство по менеджменту рисков информационной безопасности;

ж) SO/IEC 27040:2015 – «Информационные технологии. Методы обеспечения безопасности. Безопасность хранения данных». Дата выхода – 2015 год.

Организационно-технические и режимные меры и методы. Для построения политики информационной безопасности рассматривают следующие направления защиты ИС:

– защита объектов ИС;

– защита процессов, процедур и программ обработки информации;

– защита каналов связи;

– подавление побочных электромагнитных излучений;

– управление системой защиты.

Организационная защита обеспечивает:

– организацию охраны, режима, работу с кадрами, с документами;

– использование технических средств безопасности (например, простейших дверных замков, магнитных или иных карт и др.), информационно-аналитическую деятельность по выявлению внутренних и внешних угроз.

Рассмотрим основные особенности разработки политики безопасности учреждения на примере банка[7].

На первом этапе производится *изучение фактического состояния организации по обеспечению информационной безопасности*. Здесь целью является определения уязвимых мест, угроз информационной безопасности банка по следующим основным направлениям:

• организационная структура банка и нормативно-распорядительная документация в области безопасности, их соответствие целям информационной безопасности;

• система управления доступом, в том числе угрозы несанкционированного доступа посторонних на объект;

• защита речевой информации в помещениях и каналах связи, в том числе выявление естественных каналов утечки информации (оргтехника, коммуникации и т. п.), выявление искусственных каналов утечки информации (подслушивающие устройства, системы и т. п.);

---

[7] Создание систем обеспечения информационной безопасности // Официальный сайт ОАО «Центр банковских технологий» [Электронный ресурс]. 2015. URL: www.cbt.by/main.aspx?guid=1841 (дата обращения: 20.05.2014).

• безопасность компьютерных систем, в том числе проведение исследований специальными средствами сканирования безопасности в сети на предмет:

– наличия уязвимостей в оборудовании, операционных системах, прикладном программном обеспечении;

– угроз несанкционированного доступа к информации;

– угроз утечки информации при передаче по каналам связи;

– угроз нарушения целостности информации;

– угроз нарушения устойчивой работы информационной системы в целом;

– потенциальных угроз информационной безопасности со стороны работников банка при их санкционированном доступе к информационным ресурсам;

– возможности проведения атак посторонними пользователями (злоумышленниками) и др.

Ущерб от НСД (атаки) может быть представлен положительным числом в приблизительном соответствии с табл. 2.2.

Таблица 2.2

Условная численная шкала для оценки ущерба банку от НСД

Величина ущерба	Описание
0	Раскрытие информации принесет ничтожный моральный и финансовый ущерб банку (фирме)
1	Ущерб от атаки есть, но он незначителен, основные финансовые операции и положение банка на рынке не затронуты
2	Финансовые операции не ведутся в течение некоторого времени, за это время банк терпит убытки, но его положение на рынке и количество клиентов изменяются минимально
3	Значительные потери на рынке и в прибыли. От банка уходит ощутимая часть клиентов
4	Потери очень значительны, банк на период до года теряет положение на рынке. Для восстановления положения требуются крупные финансовые займы
5	Банк прекращает существование

Вероятность НСД (атаки) представляется в приблизительном соответствии с табл. 2.3.

Таблица 2.3

Вероятностно-временная шкала реализации несанкционированного доступа к информационным ресурсам

Вероятность события	Средняя частота события (НСД)
0	Данный вид атаки отсутствует
0,1	Реже, чем раз в год
0,2	Около 1 раза в год
0,3	Около 1 раза в месяц
0,4	Около 1 раза в неделю
0,5	Практически ежедневно

Далее создается таблица рисков (табл. 2.4). На этапе анализа таблицы рисков задаются некоторым максимально допустимым риском (порог), например значением 0,5.

Далее проверяется каждая строка таблицы: превышен или не превышен порог для значения риска, связанного с анализируемой атакой? Если такое превышение имеет место, данная атака должна рассматриваться с точки зрения одной из первоочередных целей разработки политики безопасности.

Таблица 2.4

Таблица рисков

Описание атаки	Ущерб	Вероятность	Риск (Ущерб · Вероятность)
Спам (переполнение почтового ящика)	1	0,4	0,4
Копирование жесткого диска из центрального офиса	3	0,1	0,3
...	...	...	...
Итого			0,9

Если интегральный риск (итого) превышает допустимый уровень, значит, в системе безопасности набирается множество мелких проблем, которые также нужно решать комплексно. В этом случае из строк таблицы (типов атак) выбираются те, которые «дают» самый значительный вклад в значение интегрального риска. Производится работа по снижению их влияния или полному устранению.

На втором этапе разрабатываются *концепция информационной безопасности* (банка, фирмы, предприятия) и направления ее практической реализации.

Концепция является базовым нормативно-информационным документом и служит основой:

- для создания единой системы правовых, организационных, технических, режимных и иных мер, обеспечивающих защищенность организации в информационной сфере;
- разработки программ и мероприятий по обеспечению информационной безопасности различных объектов, подготовки локальных нормативных документов и политик.

Объектами защиты являются все элементы информационно-технологической инфраструктуры организации (банка), а именно: помещения, компьютерное, периферийное, сетевое оборудование и каналы связи, терминалы, носители информации и программное обеспечение, данные и информация, функционирование документооборота и бизнес-процессов, внутренняя конфиденциальная информация, т. е. все элементы бизнеса, нарушение и несанкционированный доступ к которым ведут к ущербу и потерям бизнеса.

Подлежащая защите информация может находиться на бумажных носителях, в электронном виде, передаваться в виде электрических (телефон, телефакс, телекс) или акустических сигналов, записываться и воспроизводиться с помощью технических средств (диктофоны, видеомагнитофоны).

Таким образом, в каждом конкретном случае информационная безопасность должна рассматриваться системно, как комплекс инструментов по защите программно-технических или иных средств.

Информационная безопасность должна обеспечивать выполнение трех основных условий:

1) программно-технические средства должны исправно работать в соответствии с установленной конфигурацией и настройками;

2) на программно-технических средствах должно выполняться только разрешенное программное обеспечение. Любые другие программы, в том числе деструктивные, не должны попадать и активизироваться в системе;

3) доступ к внутренним ресурсам информационной системы должны иметь только авторизованные субъекты согласно своим правам.

Направления практической реализации политики безопасности банка можно представить следующим образом[8]:

- защита помещений и объектов от несанкционированного доступа;
- защита информации в бумажном документообороте банка;
- защита информации от утечки по техническим каналам, возможные угрозы при проведении мероприятий и эксплуатации технических средств, когда возможны утечки или нарушения целостности информации, нарушения работоспособности технических средств:

– побочные электромагнитные излучения информативного сигнала от технических средств;

– электрические сигналы или радиоизлучения, обусловленные воздействием на автоматизированную систему (АС) банка высокочастотных сигналов по радиоэфиру и проводам либо сигналов промышленных радиотехнических устройств (радиовещательные, радиолокационные станции, средства радиосвязи и т. п.), и модуляцией их информативным сигналом (облучение, «навязывание»);

– радиоизлучения или электрические сигналы от внедренных специальных электронных устройств перехвата информации;

– акустическое излучение информативного речевого сигнала;

– просмотр информации с экранов дисплеев и других средств ее отображения визуально и с помощью оптических средств;

– воздействие на технические или программные средства в целях нарушения целостности (уничтожения, искажения) информации;

– непреднамеренное попадание защищаемой информации к лицам, не допущенным к ней, но находящимся в пределах контролируемой зоны;

– непреднамеренное прослушивание без использования технических средств разговоров, ведущихся в выделенном помещении, из-за недостаточной звукоизоляции его ограждающих конструкций, систем вентиляции и кондиционирования воздуха;

- обеспечение безопасности информации, передаваемой по каналам связи и в системе внутреннего электронного документооборота между подразделениями банка;

---

[8] Политика информационной безопасности банка // Официальный сайт ОАО «Центр банковских технологий». [Электронный ресурс]. 2015. URL: www.cbt.by/main.aspx?guid=1841 (дата обращения: 20.05.2014).

• обеспечение информационной безопасности в компьютерных сетях;

• обеспечение информационной безопасности в платежных системах;

• обеспечение информационной безопасности в платежных системах с использованием банковских пластиковых карточек:

– меры технологического характера, имеющие целью повысить надежность процедуры передачи информации платежной системы и идентификации владельца карточки;

– меры безопасности при проведении операций с пластиковыми картами непосредственно в подразделениях, осуществляющих их выпуск, выдачу и обслуживание;

– меры безопасности при обслуживании терминального оборудования (банкоматы, инфокиоски, терминалы), включая меры по предупреждению и/или обнаружению противоправных действий с терминальным оборудованием, совершаемых третьими лицами в процессе его функционирования;

– организация распределения и закрепления за персоналом реквизитов доступа (ключей, служебных карточек операторов и администраторов, паролей, прав доступа);

• обеспечение информационной безопасности при использовании Интернет-ресурсов:

– использование сети Интернет сотрудниками банка;

– доступ к сервисам банка клиентов через Интернет (Интернет-банкинг);

– использование электронной почты, как внутрикорпоративной, так и через Интернет;

• обеспечение безопасности информации в автоматизированных системах;

• архивы электронной информации;

• размещение компьютерного и телекоммуникационного оборудования;

• регистрация пользователей и использование паролей на уровне операционных систем, прикладного программного обеспечения и компьютерной сети;

• управление доступом на уровне операционных систем, прикладного программного обеспечения и компьютерной сети;

• средства предотвращения НСД для операционных систем, прикладного программного обеспечения и компьютерной сети;

- защита от вирусов и других вредоносных программ;
- системы мониторинга безопасности операционных систем, прикладного программного обеспечения и компьютерной сети;
- системы сканирования безопасности операционных систем, прикладного программного обеспечения и компьютерной сети;
- средства аудита безопасности операционных систем, прикладного программного обеспечения и компьютерной сети;
- анализаторы протоколов, ведущихся операционными системами, прикладным программным обеспечением и устройствами компьютерной сети;
- порядок приемки-сдачи общесистемного и прикладного программного обеспечения;
- установка обновлений на общесистемное и прикладное программное обеспечение;
- резервное копирование, хранение и уничтожение носителей информации;
- обеспечение непрерывной работы и восстановления программно-технических комплексов;
- рекомендации о службах информационной безопасности и их составе на всех уровнях (головной банк, филиал, отделение).

Аппаратные, программно-аппаратные и программные способы и средства обеспечения информационной безопасности. В определенной степени этого аспекта проблемы мы уже касались выше. В общем анализируемые способы и средства условно можно классифицировать следующим образом:

1) средства защиты от несанкционированного доступа:
а) средства авторизации;
б) аудит;
2) системы мониторинга:
а) системы мониторинга сетей;
б) анализаторы протоколов;
3) антивирусные средства:
а) антивирусные программы;
б) программные и иные антиспамовые средства;
в) межсетевые экраны;
4) криптографические средства:
а) шифрование данных;
б) электронная цифровая подпись;
5) системы бесперебойного питания;

6) системы аутентификации:

а) пароль;

б) ключ доступа (физический или электронный);

в) биометрия (анализаторы отпечатков пальцев, анализаторы сетчатки глаза, анализаторы голоса, анализаторы геометрии ладони и др.).

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. К чему приводят электромагнитные или ионизирующие излучения?
2. Приведите пример источников электромагнитных или ионизирующих излучений.
3. Что такое «преднамеренная помеха», «непреднамеренная помеха»?
4. Из каких основных частей состоит периметр современной ИС?
5. Как можно осуществить неавторизованный доступ к сетевому трафику?
6. К каким последствиям могут привести отказы системы электроснабжения ИС?
7. Что такое «компьютерный вирус»? Чем он отличается от остальных деструктивных программ? Приведите примеры известных вирусов.
8. Охарактеризуйте известные Вам деструктивные программные средства.
9. Охарактеризуйте и приведите примеры критических информационных систем.
10. В чем сущность системного подхода при проектировании политики безопасности?
11. Дайте классификацию методов и средств защиты информации.
12. Перечислите основные направления реализации политики информационной безопасности.
13. Перечислите и охарактеризуйте концептуальные аспекты информационной безопасности Республики Беларусь.
14. В чем назначение организационных методов защиты информации?
15. В чем назначение правовых методов защиты информации?
16. Состав правовой защиты информации.

17. Особенности международного классификатора компьютерных преступлений.

18. В чем назначение режимных методов защиты информации?

19. Какие национальные правовые акты, регламентирующие доступ к информационным ресурсам, Вы знаете?

20. Какие международные правовые акты, регламентирующие доступ к информационным ресурсам, Вы знаете?

21. В чем заключается назначение организационно-технических методов защиты информации?

22. Как можно защитить каналы связи?

23. Как можно защититься от мешающих электромагнитных излучений?

24. Перечислите известные методы и средства авторизации.

25. Назначение межсетевых экранов.

26. Назначение источников бесперебойного питания.

27. На чем основаны биометрические средства идентификации?

Раздел II

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

Глава 3

СУЩНОСТЬ КРИПТОГРАФИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ

3.1. Основы понятия предметной области. Цели и задачи криптографии

Передача информации (данных) осуществляется между двумя абонентами, называемыми *источником сообщения* (ИсС) и *получателем сообщения* (ПС). Источником и получателем могут быть люди либо технические средства. ИсС и ПС обмениваются информацией посредством *канала передачи*. Отметим также, что и в *системах хранения информации* всегда можно выделить ИсС и ПС. В данном случае каналом передачи здесь выступает *устройство хранения информации* (память). Например, при записи данных в ОЗУ (оперативное запоминающее устройство) компьютера в качестве ИсС и ПС может выступать процессор (соответственно при записи и чтении данных). Таким образом, простейшая информационная система состоит из трех перечисленных элементов. Ее обобщенная структурная схема приведена на рис. 3.1 (здесь параметр k означает число символов в сообщении).

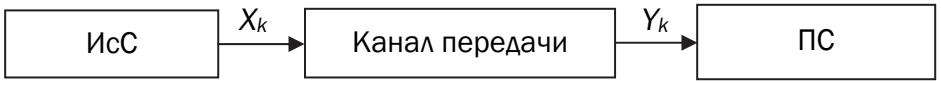


Рис. 3.1. Обобщенная структурная схема информационной системы (системы передачи информации)

Под определение ИС (ИВС) подпадает любая система обработки информации. Далее будем рассматривать ИС как совокуп-

ность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.

Математической основой описания и анализа процессов в ИС в широком смысле является теории информации. Возникновение теории информации связывают обычно с появлением фундаментальных работ К. Шеннона[9].

Определение. Построение сигнала по определенным правилам, обеспечивающим соответствие между сообщением и сигналом, несущим это сообщение, называют *кодированием*.

Кодирование в широком смысле – *преобразование сообщения в сигнал*.

Кодирование в узком смысле – *представление исходных знаков*, называемых символами, в другом алфавите.

Оно осуществляется с различными целями: повышение надежности передачи, снижение физического объема сообщения, повышение уровня конфиденциальности или безопасности. Указанные цели соответствуют *трем базовым способам преобразования сообщения* (X_k) до его передачи по каналу связи:

- кодирование или помехоустойчивое кодирование;
- сжатие или архивирование сообщений;
- криптографическое преобразование[10].

Дальнейшая часть настоящего пособия направлена на изучение и анализ методов криптографии.

В течение последних 20 лет наблюдается бурное развитие открытых академических исследований в этой области. Пока обычные граждане использовали «классическую» криптографию, «компьютерная» криптография, еще со времен Первой мировой войны, применялась исключительно в военных целях.

---

[9] См., например, Шеннон К. Работы по теории информации и кибернетике. М.: Изд-во иностранной литературы, 1963. 830 с.

[10] Сущность и примеры реализации методов, относящихся к первым двум из перечисленных классов преобразования информации, можно найти в литературе: Урбанович П. П., Шиман Д. В. Защита информации и надежность информационных систем. Минск: БГТУ, 2013. 90 с.; Урбанович П. П., Романенко Д. М., Кабак Е. В. Информационная безопасность и надежность систем. Минск: БГТУ, 2007. 90 с.

Говоря об исторических аспектах научных исследований в области криптографии, необходимо отметить тот факт, что весь период с древних времен до 1949 года можно назвать донаучным периодом, когда средства «закрытия» информации не имели строгого математического обоснования. Поворотным моментом, придавшим криптографии научность и выделившим ее в отдельное направление математики, явилась публикация уже упоминавшейся работы К. Шеннона «Теория связи в секретных системах».

Современная криптография широко используется и за стенами военных ведомств. Рядовые пользователи информационных технологий получили возможность защититься от всемогущих вредителей (не только от хакеров или кракеров). Ниже проанализируем основные особенности и оценим эффективность криптографических методов преобразования (защиты) информации.

Как отмечалось выше, *криптография* является одной из двух ветвей общего научного направления – *криптологии*. Второй ветвью криптологии является *криптоанализ*. Цели криптографии и криптоанализа прямо противоположны.

Криптографические методы нашли широкое применение в практической информатике для решения многочисленных проблем *информационной безопасности*. В проблематике современной криптографии можно выделить следующие три типа основных задач:

1) обеспечение конфиденциальности (секретности);

2) обеспечение анонимности (неотслеживаемости);

3) обеспечение *аутентификации* информации и источника сообщения.

Первый тип задач относится к защите информации от *несанкционированного доступа* по секретному *ключу*. Доступ к информации (информационным ресурсам) имеют только обладатели ключа. Второй и третий типы задач обязаны своей постановкой массовому применению электронных способов обработки и передачи информации (банковская сфера, электронная коммерция, каналы межличностной коммуникации и др.).

Криптографическое преобразование, как и два иных из числа вышеупомянутых (помехоустойчивое кодирование и сжатие), состоит из двух этапов: прямого и обратного. Прямое преобразование называют *шифрованием* или *зашифрованием* (в соответствии со стандартом ISO 7492-2 – *зашифрование*, encrypt), обратное – *дешифро-*

ванием или *расшифрованием* (*расшифрование*, decrypt). Процесс передачи зашифрованных сообщений иллюстрирует рис. 3.2.

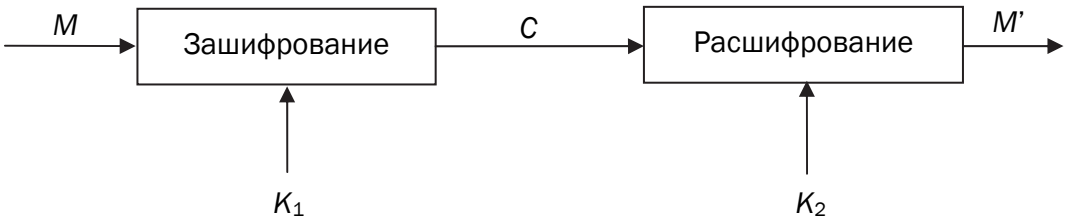


Рис. 3.2. Общая структурная схема криптосистемы

Исходное сообщение называется открытым текстом (*M*, от англ. *message*). Зашифрованное сообщение – *шифртекстом* или *шифрограммой* (*C*, от англ. *cipher*). После обратного преобразования получаем исходный (или приближенный к нему) документ (*M*'). Таким образом, в канал передается шифртекст *C*.

Если пользоваться символьными обозначениями, введенными нами ранее (X_k – данные до прямого преобразования, X_n – данные после прямого преобразования), т. е. положить $X_k = M$ и $X_n = C$,то сравнительной характеристикой криптографических методов будет равенство $k = n$. Это означает, *что длина открытого и зашифрованного сообщений не меняется* (исключение составляют методы шифрования с использованием электронной цифровой подписи).

Функция зашифрования *E* в математическом виде представляется следующим образом:

$$E(M) = C. \tag{3.1}$$

В обратном процессе функция расшифрования *D* восстанавливает *M*:

$$D(C) = M. \tag{3.2}$$

Поскольку смысл зашифрования и последующего расшифрования сообщения заключается в восстановлении исходного открытого текста, справедливо следующее равенство:

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

В этом анализе прямого и обратного процессов преобразования шифр отождествляется с *криптографическим алгоритмом*, представляющим собой математическую функцию, которая используется для зашифрования и для расшифрования информации.

До появления компьютеров криптография основывалась именно на таких алгоритмах, которые называют также текстовыми. Основой их были операции замены одних символов другими либо перестановка символов местами. Первые алгоритмы относятся к классу подстановочных, другие – перестановочных. Современные криптосистемы используют как подстановки, так и перестановки символов.

3.2. Подстановочные и перестановочные шифры

3.2.1. Подстановочные шифры

Сущность подстановочного шифрования состоит в том, что, как правило, исходный текст (*M*) или зашифрованный текст (*C*) используют один и тот же алфавит, а тайной является алгоритм подстановки. Такой шифр называется простым или *моноалфавитным*.

Примером такого шифра является известный *шифр Цезаря*, в котором каждый символ открытого текста заменяется символом, находящимся тремя символами правее: $k = 3$ (по модулю 26 или по принципу кольца): «A» меняется на «D», «B» – на «E», «W» – на «Z», «X» – на «A» и т. д. (в некоторых случаях во внимание принимается 27-й символ – пробел). Для расшифрования необходимо выполнить обратную замену. Как видим, такая криптосистема строится на основе некоторой *таблицы подстановок*.

Если сопоставить каждому символу алфавита его порядковый номер (индекс), начиная с 0, то зашифрование и расшифрование можно выразить соотношениями:

$$y = (x + k) \bmod N, \tag{3.3}$$

$$x = (y - k) \bmod N, \tag{3.4}$$

где x и y – соответственно порядковые номера (индексы) символов открытого и зашифрованного текстов; k – ключ; N – мощность алфавита (количество символов).

Пример 3.1. Имеем открытый текст M = «*cba*». На основе шифра Цезаря C = «*fed*».

Здесь $k = 3$, $N = 26$. Первый символ открытого текста (*c*) имеет индекс 2 (помним, что начальный символ алфавита (*a*) имеет ну-

левой индекс). Значит, первый символ шифртекста (c) будет иметь индекс $2 + k = 5$. А такой индекс в алфавите принадлежит символу f, и т. д.

Известное послание Цезаря *VENI VIDI VICI* (в переводе на русский означает «Пришел, увидел, победил»), направленное его другу Аминтию после победы над понтийским царем Фарнаком, выглядело бы в зашифрованном виде так:

YHQL YLGL YLFL.

Простой подстановочный шифр используется, например, в системе UNIX: простая программа шифрования (ROT13) использует смещение на 13 позиций, т. е. символ «A» заменяется на «N», и т. д.

Анализируемые шифры взламываются без труда, поскольку не скрывают частоту (вероятность) использования различных символов в открытом (а соответственно – и в зашифрованном) тексте.

Как видим, в шифре Цезаря использовались только аддитивные свойства множества целых чисел. Однако концепция, заложенная в систему шифрования Цезаря, оказалась весьма плодотворной, о чем свидетельствуют ее многочисленные модификации. Кратко проанализируем некоторые.

Применяя одновременно операции сложения и умножения по модулю n над элементами множества (индексами букв алфавита), можно получить *систему подстановок*, которую называют **аффинной системой подстановок Цезаря.** Определим преобразование в такой системе:

$$y = (a \cdot x + b) \bmod N,$$

где a и b – целые числа.

При этом взаимно однозначные соответствия между открытым текстом и шифртекстом будут иметь место только при выполнении следующих условий: $0 \le a, b < N$, наибольший общий делитель (НОД) чисел a, b равен 1, т. е. эти числа являются *взаимно простыми*.

Пример 3.2. Пусть $N = 26$, $a = 3$, $b = 5$. Тогда НОД (3, 5) = 1, и мы получаем следующее соответствие между индексами букв:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$3x + 5$	5	8	11	14	17	20	23	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2

Преобразуя числа в буквы английского алфавита, получаем следующее соответствие для букв открытого текста и шифртекста:

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

В соответствии с последней таблицей открытому тексту *BELSTU* будет соответствовать зашифрованный *IRMHKN*.

Система шифрования Цезаря с ключевым словом является одноалфавитной системой подстановки. Особенностью этой системы является использование *ключевого слова* для смещения и изменения порядка символов в алфавите подстановки (желательно, чтобы все буквы ключевого слова были различными) и некоторого числа a: $0 \le a < N$. Рассмотрим систему на примере.

Пример 3.3. Выберем некоторое число a и слово или короткую фразу в качестве *ключевого слова*. Пусть выбраны слово *DIPLOMAT* в качестве ключевого и число $a = 5$.

Ключевое слово записывается под буквами алфавита, начиная с буквы, индекс которой совпадает с выбранным числом a, как это показано ниже:

0 1 2 3 4 5 10 15 20 25
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
D I P L O M A T

Оставшиеся буквы алфавита подстановки записываются после ключевого слова в алфавитном порядке:

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

Если открытым сообщением будет *BELSTU*, то зашифрованным – *WZAHJK*.

Расшифрование сообщения производится по правилу, которое мы рассматривали на выше проанализированных примерах.

Возникает вопрос: а что будет, если применить несколько ключей при зашифровании/расшифровании одного сообщения? Тогда получится система, известная как *шифр Виженера*[11].

[11] Французский дипломат Блез де Виженер (фр. *Blaise de Vigenère)* представил описание простого, но стойкого шифра перед комиссией Генриха III во Франции в 1586 году.

В **шифре Виженера**, как было отмечено, мы имеем дело с последовательностью сдвигов, циклически повторяющейся. Основная идея заключается в следующем. Создается таблица (*таблица Виженера*) размером $N \times N$ (N – число знаков в используемом алфавите). Эти знаки могут включать не только буквы, но и, например, пробел или иные знаки. В первой строке таблицы записывается весь используемый алфавит. Каждая последующая строка получается из предыдущей циклическим сдвигом последней на 1 символ влево. Таким образом, при мощности алфавита (английского языка), равной 26, необходимо выполнить последовательно 25 сдвигов для формирования всей таблицы.

Вид такой таблицы (*квадрата Виженера*) представлен на рис. 3.3.

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

Рис. 3.3. Таблица (квадрат) Виженера

Шифрование происходит на основе формулы (3.3). Выбор символа подстановки при зашифровании каждого символа сообщения (выбираются из крайнего левого столбца таблицы) происходит определением пересечения этого символа и соответст-

вующего ему (с тем же индексом) символа ключевой последовательности (выбирается из верхней строки таблицы). Например, если первым символом сообщения будет символ *B*, а первым символом ключа будет символ *T*, то первым символом шифртекста будет символ *U* (находится на пересечении 2-й строки (*B*) и 20-го столбца (*T*) таблицы): здесь $k = 18$; при зашифровании символа *E* выбирается 5 строка, и если вторым символом ключа будет *I* (9-й столбец матрицы), то вторым символом шифртекста будет *M*: здесь уже $k = 4$. При расшифровании следует использовать формулу (3.4).

Именно такая таблица, а не одна строка, как в шифре Цезаря, используется для зашифрования/расшифрования сообщения. Таким образом, в определенном смысле рассматриваемый шифр состоит из 25 шифров Цезаря. И при зашифровании каждого иного символа сообщения используется иное значение *k*. Важно подчеркнуть, что длина шифруемого сообщения и длина ключа должны быть одинаковыми. Процедуры зашифрования/расшифрования также проанализируем с помощью примера.

Пример 3.4. Предположим, что шифруемое сообщение имеет вид *M* = «*BELSTU*». Как и в *системе шифрования Цезаря с ключевым словом*, в нашем случае необходимо выбрать ключ. Пусть ключом будет слово «*TIR*». Желательно выбирать ключевое выражение, имеющее длину сообщения с минимальными повторениями одинаковых символов, что влияет на криптостойкость шифра при вероятностном криптоанализе. Если же, как в нашем случае, ключ имеет меньшую длину (3 символа в сравнении с 6), то его следует дополнить путем циклического повторения. Для нашего примера после указанной операции полное содержание ключевой информации будет иметь вид *TIRTIR*.

Запишем сообщение, ключ и шифртекст в виде таблицы:

Сообщение	**B**	**E**	**L**	**S**	**T**	**U**
Ключ	**T**	**I**	**R**	**T**	**I**	**R**
Шифртекст	**U**	**M**	**C**	**L**	**B**	**L**

Даже из этого простого примера видно, что разным буквам сообщения (*S*, *U*) соответствует одна и та же буква в шифртексте (*L*).

Следует отметить, что для выполнения эффективного частотного (вероятностного) криптоанализа аналитику важно знать дли-

ну ключа, если его начальная длина (как в последнем примере) меньше длины шифруемого сообщения.

3.2.2. Перестановочные шифры

Перестановочные шифры используют перестановку символов исходного сообщения в соответствии с установленным правилом. Открытый текст остается неизменным, но символы в нем «перетасовываются» (подвергаются *пермутации*). Так, в *простом вертикальном перестановочном шифре* открытый текст пишется по горизонтали на разграфленном листе бумаги фиксированной длины, а шифртекст считывается по вертикали. Рассмотрим это на примере.

Пример 3.5. *М* = «ВАСЯ ЛЮБИТ МАШУ». Запишем этот текст как показано на рис. 3.4.

В	А	С	Я	_
Л	Ю	Б	И	Т
_	М	А	Ш	У

Рис. 3.4. Использование простого вертикального перестановочного шифра

Считывание по столбцам снизу вверх приводит к такому шифртексту: *С* = «_ЛВМЮААБСШИЯУТ_».

Принцип записи исходного сообщения и порядок считывания символов может быть различным. Обратимся к следующему примеру.

Пример 3.6. Открытый текст возьмем из предыдущего примера, а запишем его так, как показано на рис. 3.5.

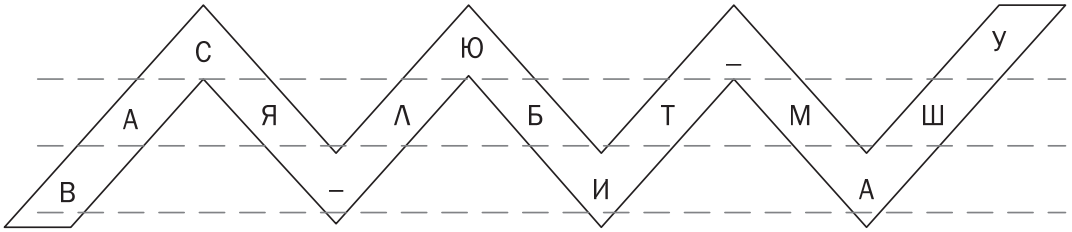


Рис. 3.5. Использование перестановочного шифра

Осуществляя считывания по уровням, начиная с верхнего, получим следующий шифртекст: *C* = «СЮ_УАЯЛБТМШВ_ИА».

Существуют еще более сложные перестановочные шифры, но компьютеры достаточно быстро справляются с ними. При этом использование данных шифров требует большого объема памяти.

Если защита, обеспечиваемая алгоритмом, основана на сохранении в тайне самого алгоритма, то это ограниченный алгоритм. Ограниченные алгоритмы представляют некоторый исторический интерес, но не соответствуют современным стандартам.

Ограниченные алгоритмы не допускают эффективного контроля или стандартизации. Каждая группа пользователей должна использовать собственный уникальный алгоритм. Такие группы не могут использовать открытые аппаратные или программные продукты – злоумышленник может приобрести такой же продукт и раскрыть алгоритм. Этим группам приходится разрабатывать и реализовывать собственные алгоритмы.

Несмотря на указанные фундаментальные недостатки, ограниченные алгоритмы необычайно популярны в приложениях с низким уровнем защиты. Пользователи либо не осознают проблем, связанных с безопасностью своих систем, либо слабо заботятся о решении проблемы.

3.3. Симметричные и асимметричные шифры

Современная криптография решает проблему с помощью ключа, который на рис. 3.2 обозначен буквой *K*. Такой ключ может быть любым значением, выбранным из большого множества. Множество возможных ключей называют *пространством ключей*.

Ключ – секретный параметр, управляющий ходом преобразования. Ключ определяет конкретный вариант преобразования. Ключ используется в обеих операциях: как зашифрования, так и расшифрования. Таким образом, теперь функции зашифрования и расшифрования принимают следующий вид:

функция зашифрования *E*:

$$E_K(M) = C, \tag{3.5}$$

функция расшифрования *D*:

$$D_K(C) = M \tag{3.6}$$

или

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

Соотношения (3.5)–(3.6) описывают криптосистемы, использующие одинаковое значение ключа для зашифрования и расшифрования ($K_1 = K_2 = K$). Такие криптосистемы называются *симметричными*.

Простейшим примером симметричного криптопреобразования является сложение по модулю двух сообщений M с ключом K для получения шифртекста. Расшифрование достигается выполнением такой же операции над C и K.

Симметричные криптосистемы называются также криптосистемами с *тайным ключом*, поскольку значение ключа должно быть известно только отправителю и получателю сообщений.

Другой класс современных криптоалгоритмов для зашифрования и расшифрования использует различные ключи ($K_1 \neq K_2$).

В этом случае формальное представление анализирумых преобразований представляется следующим образом:

функция зашифрования E:

$$E_{K_1}(M) = C, \tag{3.7}$$

функция расшифрования D:

$$D_{K_2}(C) = M \tag{3.8}$$

или

$$D_{K_2}(E_{K_1}(M)) = M. \tag{3.9}$$

Ключи K_1 и K_2 являются разными, но взаимозависимыми (один из них тайной не является). Поэтому асимметричные криптосистемы называют также криптосистемами с открытым или публичным ключом.

3.4. Блочные и потоковые шифры

Симметричные алгоритмы подразделяются на два подкласса. Одни алгоритмы обрабатывают открытый текст побитово (иногда побайтово). Такие алгоритмы называют *потоковыми*. Другие алгоритмы обрабатывают группы (блоки) битов открытого текста. Эти алгоритмы называют *блочными*.

В современных компьютерных алгоритмах типичный размер блока составляет 64 бита. Более подробное рассмотрение блочных и потоковых (иногда в литературе их называют поточными) шифров будет дано ниже.

3.5. Особенности криптоанализа

Предназначение криптографии заключается в сохранении в тайне открытого текста или ключа (или того и другого). Предполагается, что *интруз* располагает неограниченным доступом к каналам связи. *Интруз* – физическое лицо или процесс, которые реализуют неразрешенный или несанкционированный доступ к информации (*атаку* на систему).

Задачей криптоанализа является восстановление открытого текста без доступа к ключу. Попытка криптоанализа также называется *атакой*. А раскрытие ключа без привлечения специальных методов называют *компрометацией ключа*.

Надежность или криптостойкость симметричных и асимметричных алгоритмов зависит от ключей, а не от самих алгоритмов.

Голландский криптограф Август Керкгоффс[12] (1835–1903) впервые сформулировал постулаты оценки *стойкости шифра* (или *криптостойкости*) перед его *взломом* (раскрытием), в соответствии с которыми:

1) весь механизм преобразования считается известным злоумышленнику (*интрузу*);

2) *криптостойкость* (надежность) алгоритмов преобразования определяется только неизвестным значением ключа.

В современном *криптоанализе* (вспомним, что целью криптоанализа является взлом шифра) рассматриваются атаки на засекречивающие системы на основе следующих известных данных:

1) шифртекста;

2) открытого текста и соответствующего ему шифртекста;

3) выбранного открытого текста;

4) выбранного шифртекста;

5) на основе подобранного ключа.

---

[12] Kerckhoffs A. La cryptographie militaire // Journal des sciences militaires. Vol. IX. Jan. 1883, p. 5–38; Feb. 1883, p. 161–191.

В случае *атаки на основе шифртекста* криптоаналитик располагает шифртекстами нескольких сообщений, зашифрованных одним алгоритмом. Его задача состоит в расшифровании сообщений либо в определении ключа (предпочтительно).

При *атаках на основе открытого текста* криптоаналитик располагает шифртекстами нескольких сообщений и открытыми текстами этих же сообщений. Его задача – определить ключ.

Атака на основе выбранного открытого текста предполагает не только возможность доступа криптоаналитика к шифртекстами нескольких сообщений и открытым текстам этих же сообщений, но и возможность выбирать открытый текст для зашифрования. Задача – определить ключ.

Атака на основе выбранного шифртекста позволяет криптоаналитику выбирать различные шифртексты для расшифрования. Он также имеет доступ к расшифрованным текстам. Атака применяется, главным образом, на асимметричные алгоритмы. Задача – определить ключ.

В случае *атаки на основе подобранного ключа* криптоаналитик кое-что знает о связях между ключами.

Сложность той или иной атаки можно оценит по следующим критериям:

по сложности данных – оценивается объем данных, необходимых для реализации атаки;

по сложности обработки – оценивается время, необходимое на реализацию атаки (фактор трудозатрат);

по требованиям к памяти компьютера – оценивается минимально необходимый объем памяти компьютера для выполнения всех расчетно-аналитических операций.

Ларс Кнудсен (Lars Knudsen) классифицировал сложность взлома алгоритмов по нескольким критериям:

полное вскрытие – криптоаналитик находит ключ, такой что $D_K(C) = M$ (см. формулу (3.6));

глобальная дедукция – криптоаналитик находит альтернативный алгоритм, эквивалентный формуле (3.6) без знания K;

случайная (или частичная) дедукция – криптоаналитик находит (или крадет) открытый текст для перехваченного зашифрованного варианта;

информационная дедукция – криптоаналитик добывает некоторую информацию о ключе или об открытом тексте (несколько битов ключа или фрагменты открытого текста).

Криптографический алгоритм считается *безусловно стойким*, если восстановление открытого текста невозможно при любом объеме шифтекста.

Такой алгоритм реализован в *шифре Вернама*, или *шифре на основе одноразовых блокнотов*. Данный шифр предложили в 1917 году Гилберт Вернам (Gilbert Verman) и Мэйджор Моборн (Major Mauborgne). Классический одноразовый блокнот – это неповторяющийся случайный набор ключей. Каждый ключ используется только один раз и только в одном сообщении. Вторая важная особенность – длина ключа должна быть не меньше длины шифруемого сообщения.

Все остальные криптосистемы можно вскрыть с использованием только шифртекста простым перебором возможных ключей и проверкой осмысленности полученного открытого текста. Такой метод называется *лобовой атакой* (от англ. *brute-force*). Время, необходимое для любового вскрытия системы, зависит от двух параметров: числа тестируемых ключей и времени тестирования отдельной ключевой комбинации.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Охарактеризуйте обобщенную структурную схему информационной системы. В чем состоит ее отличие от ИС с криптографическим преобразованием информации?
2. Сформулируйте основную задачу криптографии.
3. Охарактеризуйте основные элементы криптосистемы.
4. Дайте классификации шифров по различным признакам.
5. Создайте собственный подстановочный шифр и оцените его криптостойкость.
6. Создайте таблицу Виженера для алфавита русского (белорусского) языка. По указанию преподавателя (или самостоятельно) выберите ключ и зашифруйте сообщение, состоящее из собственных фамилии, имени и отчества.
7. Создайте таблицы, как в предыдущем задании, дополнив алфавит знаком пробела, и выполните задание 6.
8. Создайте собственный перестановочный шифр и оцените его криптостойкость.
9. На основе каких данных строятся атаки на засекречивающие системы?
10. Чем определяется сложность взлома криптоалгоритмов?

Глава 4
ОСНОВЫ КРИПТОГРАФИИ

При разработке и анализе алгоритмов шифрования используется ряд понятий теории чисел и модулярной арифметики. Ниже кратко проанализируем некоторые базовые положения этих направлений алгебры.

4.1. Элементы теории чисел

Общие сведения. Теория чисел, или высшая арифметика, изучает свойства целых чисел. К *целым* относятся, например, числа …, –4, –3, –2, –1, 0, 1, 2, 3, 4, … . *Натуральные числа* – целые положительные числа: 1, 2, 3, 4, … .

Некоторое число a является *делителем* другого числа b, если $b = a \cdot c$ для некоторого числа c. Примем обозначение $a|b$, означающее, что a делит b нацело, или a является делителем b. Если число a не является делителем числа b, то говорим: a не делит b.

Натуральное число p называется *простым*, если $p > 1$ и не имеет положительных делителей, отличных от 1 и p. Простое число не делится без остатка ни на одно другое число. Например, простыми являются числа 2, 3, 5, 73, 2521, 2 365 347 734 339 и $(2^{756\,839} - 1)$. Количество простых чисел бесконечно велико.

Натуральное число N называется *составным*, если $N > 1$ и имеет, по крайней мере, один положительный делитель, отличный от 1 и N. Единица не является ни простым, ни составным числом.

Основная теорема арифметики. Всякое натуральное число N, кроме 1, можно представить как произведение простых множителей:

$$N = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_n, n > 1.$$

Целое число 37 – простое. Целое число 1 554 985 071 = 3 · 3 × × 4463 · 38 713 – произведение четырех простых чисел, два из которых совпадают. Или 39 616 304 = 2 · 13 · 7 · 2 · 23 · 13 · 2 · 13 · 2 · 7 = = 2 · 2 · 2 · 2 · 7 · 7 · 13 · 13 · 13 · 23.

Общий делитель нескольких целых чисел – это число, на которое делятся все данные числа без остатка. Наибольший из делителей называется *наибольшим общим делителем* (НОД).

Один из способов вычисления НОД двух чисел базируется на *алгоритме Евклида*.

Пусть даны два числа – a и b; $a > 0$, $b > 0$, считаем, что $a > b$. Находим ряд равенств:

$$\left.\begin{array}{ll} a = b \cdot q_1 + r_1, & 0 < r_1 < b; \\ b = r_1 \cdot q_2 + r_2, & 0 < r_2 < r_1; \\ r_1 = r_2 \cdot q_3 + r_3, & 0 < r_3 < r_2; \\ r_2 = r_3 \cdot q_4 + r_4, & 0 < r_4 < r_3; \\ r_{n-3} = r_{n-2} \cdot q_{n-1} + r_{n-1}, & 0 < r_{n-1} < r_{n-2}; \\ r_{n-2} = r_{n-1} \cdot q_n + r_n, & 0 < r_n < r_{n-1}; \\ r_{n-1} = r_n \cdot q_{n+1}, & r_{n+1} = 0, \end{array}\right\} \quad (4.1)$$

заканчивающийся, когда получаем некоторое $r_{n+1} = 0$. Тогда r_n – наибольший общий делитель чисел a и b.

Последнее неизбежно, так как ряд b, r_1, r_2, …, как ряд убывающих целых, не может содержать более чем b положительных. Имеем: $b > r_1 > r_2 > ... > r_n > 0$, следовательно, процесс оборвется максимум через b шагов.

Пример 4.1. Пусть $a = 525$, $b = 231$. Найти НОД.

Применим алгоритм Евклида:

$$525 = 231 \cdot 2 + 63;$$
$$231 = 63 \cdot 3 + 42;$$
$$63 = 42 \cdot 1 + 21;$$
$$42 = 21 \cdot 2.$$

Получаем последний положительный остаток $r_3 = 21$.

Таким образом, НОД (525, 231) = 21.

Пример 4.2. Пусть $a = 1234$, $b = 54$. Найти НОД.

$$1234 = 54 \cdot 22 + 46;$$
$$54 = 46 \cdot 1 + 8;$$
$$46 = 8 \cdot 5 + 6;$$
$$8 = 6 \cdot 1 + 2;$$
$$6 = 2 \cdot 3.$$

Последний ненулевой остаток равен 2, поэтому НОД (1234, 54) = 2.

Схема алгоритма приведена на рисунке.

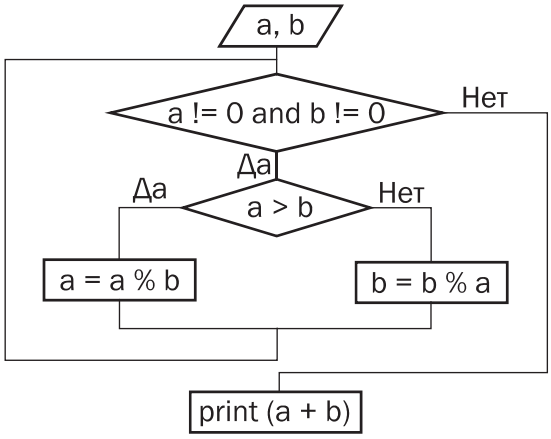


Рис. 3.5. Блок-схема алгоритма Евклида для вычисления НОД

Взаимно простые числа – это два или несколько целых чисел, наибольший общий делитель которых равен единице. Таким образом, если НОД чисел *p* и *q* равен 1, то эти числа называются взаимно простыми. Например, числа 13 и 28 являются взаимно простыми, хотя число 28 не относится к числу простых. Числа 15 и 27 взаимно простыми не являются. Простое число взаимно просто со всеми другими числами, кроме чисел, кратных данному простому числу.

Количество натуральных чисел, меньших некоторого числа *n* и взаимно простых с ним, можно подсчитать на основе известной *функции Эйлера* (по имени швейцарского математика Леонарда Эйлера (1707–1783)), иногда называемой «*фи-функцией*», φ(*n*). Например, для числа 24 (*n* = 24) существует 8 взаимно простых с ним чисел (1, 5, 7, 11, 13, 17, 19, 23), поэтому φ (24) = 8.

Если *n* – простое число, то

$$\varphi(n) = n - 1. \qquad (4.2)$$

Другие примеры:

φ(1) = 1;	φ(5) = 4;	φ(9) = 6;
φ(2) = 1;	φ(6) = 2;	φ(10) = 4;
φ(3) = 2;	φ(7) = 6;	φ(11) = 10;
φ(4) = 2;	φ(8) = 4;	φ(12) = 4.

Любое положительное целое число p может быть выражено с помощью положительных целых чисел, не превосходящих и взаимно простых с каждым делителем числа p. Например, $6 = 2 \cdot 3$ имеет четыре делителя: 1, 2, 3 и 6.

$$\varphi(1) + \varphi(2) + \varphi(3) + \varphi(6) = 1 + 1 + 2 + 2 = 6.$$

Если $n = p \cdot q$, то

$$\varphi(n) = (p - 1) \cdot (q - 1). \qquad (4.3)$$

Если числа p и q – взаимно простые, то

$$\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q).$$

Например, пусть $p = 8$ и $q = 15$. Тогда $\varphi(8) = 4$, поскольку только 1, 3, 5 и 7 – положительные целые числа, которые меньше 8 и взаимно простые с 8. Также $\varphi(15) = 8$, поскольку только 1, 2, 4, 7, 8, 11, 13 и 14 – положительные целые числа, которые меньше 15 и взаимно простые с 15. Следовательно,

$$\varphi\,(120) = \varphi\,(8) \cdot \varphi\,(15) = 32,$$

что можно проверить непосредственно.

С другой стороны, если p и q – очень большие простые числа и известен результат их перемножения (число n), то обратная задача – найти p и q по известному n (*задача факторизации*) – даже для современных вычислительных средств представляется практически неразрешимой. Эта особенность используется, в частности, в некоторых алгоритмах асимметричной криптографии.

4.2. Основы модулярной арифметики

Понятие «*модулярная арифметика*» ввел немецкий ученый К. Ф. Гаусс. В этой арифметике мы интересуемся остатком от деления числа a на число n. Если таким остатком является число b, то можно записать:

$$a \equiv b \bmod n.$$

В дальнейших математических выкладках вместо знака «≡» будем употреблять знак простого равенства: «=».

Такая формальная запись читается как «a сравнимо с b по модулю n».

При целочисленном (в том числе и нулевом) результате k деления числа a на число n справедливо $a = b + k \cdot n$.

Пример 4.3. При $a = 13$ и $n = 4$ имеем $b = 1$, т. е. $13 = 1 + 3 \cdot 4$. Для данного примера справедлив вывод: число 13 по модулю 4 равно 1 или числа 13 и 1 равны по модулю 4.

Пример 4.4. При $a = 13$ и $n = 2$ имеем $b = 1$; с другой стороны, при $a = 40$ и $n = 2$ получим $b = 0$.

Иногда b называют *вычетом* по модулю n.

Модулярная арифметика так же коммутативна, ассоциативна и дистрибутивна, как и обычная арифметика.

Приведение каждого промежуточного результата по модулю n дает такой же результат, как приведение всего результата вычисления по модулю n:

$$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n;$$

$$(a - b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n;$$

$$(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n;$$

$$(a \cdot (b + c)) \bmod n = (((a \cdot b) \bmod n) + ((a \cdot c) \bmod n)) \bmod n.$$

Модулярная арифметика, как видим, ограничивает диапазон промежуточных и конечного результатов вычислений, т. е. эти вычисления проще организовать и выполнить на компьютере.

Вычисление степени некоторого числа по модулю другого числа представляет собой последовательность операций умножения и деления. Однако существуют методы ускорения таких вычислений.

Проиллюстрируем это простыми примерами.

Пример 4.5. Нужно вычислить модуль n некоторого числа a в 8 степени: $a^8 \bmod n$.

Понятно, что семь операций умножения числа a могут дать огромное число. Порядок чисел, которыми оперирует вычислитель, можно значительно уменьшить, если воспользоваться промежуточными вычислениями по модулю: $((a^2 \bmod n)^2 \bmod n)^2 \bmod n$.

Пример 4.6. Предположим, что показатель степени не является степенью 2. Пусть это будет, например, 25, т. е. необходимо вычислить $a^{25} \bmod n$.

После понятных рассуждений последовательность операций можем представить в виде следующей *аддитивной цепочки*:

$$a^{25} \bmod n = (a \cdot a^{24}) \bmod n = ((((a^2 \cdot a)^2)^2)^2\, a) \bmod n =$$

$$= (((((((a^2 \bmod n)\, a) \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n)a) \bmod n.$$

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Приведите примеры простых и взаимно простых чисел, двоичное представление которых состоит из не менее чем 10 бит.
2. Сформулируйте основную теорему арифметики.
3. Найдите НОД пар чисел: 315 и 196; 294 и 7; 1443 и 143; 2187 и 343.
4. В чем сущность задачи факторизации?
5. Справедливо ли равенство:

$$\varphi(120) = \varphi(8) \cdot \varphi(15) = 32?$$

Поясните ответ.

6. Представьте в виде аддитивной цепочки вычисление вида:

2325 mod 12; 834 mod 23; 4531 mod 100; (145 · 964) · 5 mod 11.

4.3. Обратные значения чисел в модулярной арифметике

Вспомним, что обратное значение числа 4 есть ¼. Это означает, что их произведение должно равняться 1. В модулярной арифметике обратное значение является понятием более сложным.

Вспомним, что в анализируемой арифметике запись

$$a \cdot x \equiv 1 \bmod n$$

эквивалентна поиску таких значений x и k, которые удовлетворяли бы тождеству:

$$a \cdot x = n \cdot k + 1.$$

Общая задача вычисления обратного значения по модулю формируется следующим образом: нужно найти такое значение x, которое бы удовлетворяло уравнению:

$$(a \cdot x) \bmod n = 1. \qquad (4.4)$$

Последнюю формулу можно представить в таком виде:

$$x^{-1} = a \bmod n. \qquad (4.5)$$

Число x^{-1} является обратным значением по модулю n числа a.

Уравнения (4.4) и (4.5) могут иметь либо не иметь решения.

Пример 4.7. Необходимо найти обратное значение числа 5 по модулю 14.

Перепишем необходимое уравнение в форме (4.4):

$$(5 \cdot x) \bmod 14 = 1.$$

Легко установить, что $x = 3$, т. е. справедливо

$$3^{-1} = 5 \bmod 14.$$

С другой стороны, при $a = 2$ и при $n = 14$ уравнение решения не имеет.

Следует запомнить: 1) уравнения (4.4) и (4.5) имеют единственное решение, если числа *x* и *n* являются взаимно простыми; 2) если *n* – простое число, то любое число от 1 до *n* – 1 является взаимно простым с *n* и имеет только одно обратное значение по модулю *n*.

Обратное значение по модулю можно вычислить, воспользовавшись *расширенным алгоритмом Евклида*. Ниже (листинг) приводится реализация этого алгоритма на языке C++[13].

```
#define isEven(x) ( (x & 0x01) == 0)
# define isOdd(x) (x & 0x01)
# define swap(x,y) (x^= y, y^= x, x^= y)
void ExtBinEuclid(int *u, int *v, int *ul, int *u2, int *u3) {
//предупреждение: u и v будут переставлены, если  u < v
int k, tl, t2, t3;
if (*u < *v) swap(*u,*v);
for (k = 0; isEven(*u) && isEven(*v); ++k) {
            *u >>= 1; *v >>= 1 ;
}
*ul = 1; *u2 = 0; *u3 = *u; tl = *v; t2 = *u - 1; t3 = *v;
do {
do {
if (isEven(*u3)) {
if (isOdd(*ul) || isOdd(*u2)) {
*ul += *v; *u2 += *u;
}
```

[13] Алгоритм заимствован из книги: Шнайер Б. Прикладная криптография. М.: Триумф, 2003. 816 с.

```
                                        *ul >>= 1; *u2 >>= 1; *u3 >>= 1;
}
if (isEven(t3) || *u3 < t3) {
swap(*ul,tl); swap(*u2, t2); swap(*u3, t3);
}
} while (isEven(*u3));
while (*ul < tl || *u2 < t2) {
            *ul += *v; *u2 += *u;
}
*ul -= tl; *u2 -= t2; *u3 -= t3;
} while (t3 > 0);
while (*ul >= *v && *u2 >= *u) {
*ul -= *v; *u2 -= *u;
                                }
            *ul <<= k; *u2 <<= k; *u3 <<= k;
}
main(int argc, char  **argv) {
int a, b, gcd;
if (argc < 3) {
cerr << "Использование: xeuclid u v" << endl;
return -1;
            }
int u = atoi(argv[l]);
int v = atoi(argv[2J);
 if (u <= 0 || v <= 0 ) {
cerr << "Аргументы должны быть положительными! " << endl;
 return -2;
}
// предупреждение: u и v будут переставлены, если u < v
ExtBinEuclid(&u, &v, &a, &b, &gcd);
cout << a <<" * " << u << " + (-"
<< b << ") * << " << v << " = " << gcd << endl;
                    if (gcd == 1)
cout << "Обратное значение " << v << " mod " << u << " равно
                    "
                                << u - b << endl;
return 0;
}
```

Китайская теорема об остатках. **В общем случае, если разложение числа *n* на простые множители представляет собой $p_1 \cdot p_2 \cdot \ldots \cdot p_t$ (некоторые простые числа могут встречаться несколько раз), то система уравнений**

$$x \bmod p_i = a_i, \text{ где } i = 1, 2, \ldots, t \qquad (4.6)$$

имеет единственное решение: *x*, меньшее *n*.

Иными словами, число (меньшее, чем произведение нескольких простых чисел) однозначно определяется своими вычетами по модулю от этих простых чисел.

Китайской теоремой об остатках можно воспользоваться для решения полной системы уравнений в том случае, если известно разложение числа n на простые множители. Основной вариант этой теоремы (доказательство опущено) сформулирован в I веке н. э. китайским математиком Сун Цзе.

Пример 4.8. Положим, $n = 15$ и $p_1 = 3$, $p_2 = 5$; $a_1 = 2$, $a_2 = 4$.

Запишем систему уравнений:

$$x \bmod p_1 = a_1,$$
$$x \bmod p_2 = a_2;$$

или

$$x \bmod 3 = 2,$$
$$x \bmod 5 = 4.$$

Этот пример показывает, что существует единственное число, меньшее чем $3 \cdot 5 = 15$, с такими вычетами: 14, т. е. два вычета однозначно определяют число.

Малая теорема Ферма. Если *n* – простое число, а число *a* не кратно *n*, то справедливо

$$a^n = 1 \bmod n. \qquad (4.7)$$

Автором этой теоремы (доказательство опустим) является французский математик Пьер де Ферма (1601–1665).

В соответствии с *обобщением Эйлера* приведенной теоремы, если НОД $(a, n) = 1$, то справедливо:

$$a^{\varphi(n)} \bmod n = 1. \qquad (4.8)$$

Последнее выражение можно переписать в следующем виде:

$$a^{-1} \bmod n = a^{\varphi(n)-1} \bmod n. \qquad (4.9)$$

Пример 4.9. Найти число, обратное 5 по модулю 7.

Так как число 7 является простым, то используя выражение (4.2), получим: $\varphi(7) = 7 - 1 = 6$. Теперь с помощью формулы (4.9) получаем: $5^{6-1} \bmod 7 = 5^5 \bmod 7 = 3$.

Таким образом, $5^{-1} \bmod 7 = 3$ или $5 \cdot 3 = 1 \bmod 7$.

4.4. Проблема дискретного логарифма

Если известны три некоторых числа (a, x, n), то достаточно легко можно вычислить число y:

$$y = a^x \bmod n. \qquad (4.10)$$

Обратная задача: найти x, если известны a, y, n. Эта задача решается гораздо труднее. Ее называют *задачей (проблемой) дискретного логарифмирования*, по аналогии с вещественными числами, для которых $x = \log_a y$.

Решения существуют не для всех дискретных логарифмов (напомним, что речь идет только о целочисленных решениях).

Рассматриваемые вычисления относятся к числу так называемых *однонаправленных функций*.

Однонаправленная функция – одно из центральных понятий в асимметричной криптографии.

Наглядным примером однонаправленной функции может служить разбиение чашки: разбить чашку на мелкие кусочки достаточно просто, однако очень не просто собрать чашку из кусочков.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Сформулируйте китайскую теорему об остатках.
2. Представьте числа 9, 21, 25, 27, 33, 45, 55, 75 в виде произведения простых чисел и, используя вражение (4.6), найти решение системы уравнений $x \bmod p_i = a_i$.
3. Сформулируйте и поясните малую теорему Ферма.
4. Найдите число, обратное x по модулю n, если x и n соответственно равны: 4 и 11; 4 и 5; 7 и 3; 19 и 13; 7 и 17.
5. Поясните сущность проблемы дискретного логарифмирования.

Глава 5
ХАРАКТЕРИСТИКИ И РЕАЛИЗАЦИЯ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

5.1. Алгоритм DES

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

Для понимания существа анализируемого алгоритма целесообразно рассмотреть следующий пример.

Пример 5.1. Пусть открытое сообщение ($M = \{m_i\}$) в двоичной форме имеет вид: M = 10101100. Считаем, что выбран симметричный ключ $K_1 = K_2 = K$ = 1010. Используется самая простая операция зашифрования:

$$c_i = m_i \oplus K,$$

и операция расшифрования:

$$m_i = c_i \oplus K,$$

где c_i – i-й блок зашифрования; m_i – i-я часть сообщения.

Нетрудно убедиться, что $C = c_1 c_2 = 00000110$, а сложение каждых четырех бит шифртекста с ключом восстанавливает исходное сообщение (здесь $\oplus$ – операция суммирования по модулю 2).

Алгоритм DES (Data Encryption Standard), как и другие симметричные (и асимметричные) алгоритмы, использует множественные арифметическо-логические преобразования исходного текста.

Стандарт шифрования DES был разработан в 1970-х годах Национальным институтом стандартизации США (ANSI) и называется алгоритмом DEA (Data Encryption Algorithm).

Основные идеи алгоритма были предложены компанией IBM еще в 1960-х годах и базировались на идеях, описанных Клодом Шенноном в 1940-х годах.

DEA является блочным алгоритмом и оперирует с блоками данных размером 64 бита. При этом используется ключ длиной 56 битов; дополнительно к ним вычисляются 8 битов четности:

8-й бит в каждом из 8 байтов ключа. Такая длина ключа соответствует 10^{17} комбинаций, что обеспечивало до недавнего времени достаточный уровень безопасности.

Входной блок данных, состоящий из 64 битов, преобразуется в выходной блок идентичной длины. В алгоритме широко используются *рассеивания* (*подстановки*) и *перестановки* битов текста. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый *раундом* или *циклом*. Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов. После первоначальной перестановки и разделения 64-битного блока данных на правую (R) и левую (L) половины длиной по 32 бита выполняются 16 раундов одинаковых действий (рис. 5.1).

Рис. 5.1. Общая схема алгоритма DES

В табл. 5.1 показан принцип первоначальной перестановки входного 64-битного слова.

Выполненная перестановка означает, например, что первый бит входного блока сообщения будет размещен на 40-й позиции, а 58-й – на 1-й и т. д. Из беглого анализа выполненной перестановки легко понять принцип. Алгоритм перестановки разрабатывался

для облегчения загрузки блока входного сообщения в специализированную микросхему.

Таблица 5.1

Начальная перестановка[14]

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Поскольку программная реализация рассмотренной перестановки достаточно трудна, и с учетом того что сама процедура не влияет на общую криптостойкость алгоритма, во многих программных приложениях эта перестановка, как и конечная, из алгоритма исключается.

Для выполнения упомянутых действий вводится функция *f* (на рис. 5.1 – *f*-функция), которая оперирует с 32-разрядными словами исходного текста и использует в качестве параметра 48-разрядный ключ (в каждом из фиксированного множества преобразований биты ключа сдвигаются и затем из 56 битов ключа выбираются 48).

Схема работы функции *f* показана на рис. 5.2.

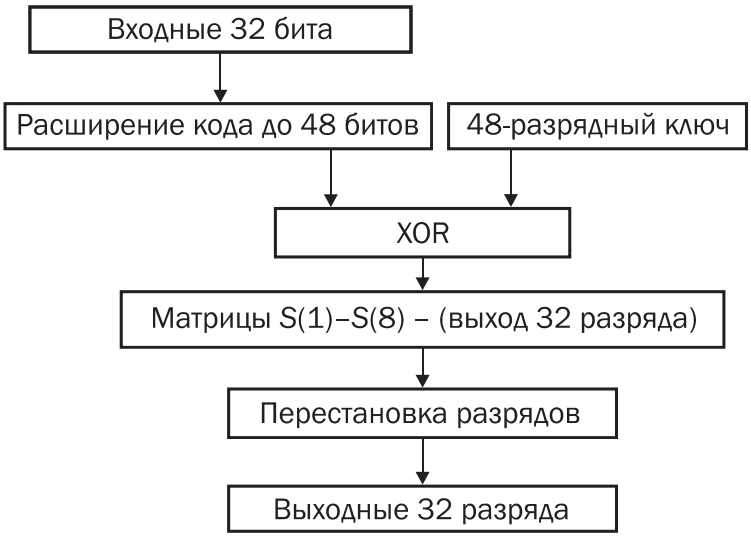


Рис. 5.2. Схема реализации функции *f*

[14] Последовательность выходных бит здесь и в последующих таблицах соответствует числам, читаемым по строкам, затем – по столбцам.

Сначала 32 входных разряда расширяются до 48 (расширяющая перестановка, при этом некоторые разряды повторяются), после чего происходит сложение их по модулю два (XOR) с ключом (по аналогии с примером 5.1):

$$\left.\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} + f(R_{i-1}, K_{i-1}). \end{aligned}\right\} \qquad (5.1)$$

Расширяющая перестановка решает две задачи: 1) приведение размера правой половины блока данных в соответствие с размером ключа; 2) получение более длинного результата (длину позднее можно изменить).

Основной криптографический смысл такого преобразования (удлинения) состоит в том, что увеличивается влияние одного бита сообщения на общий результат. Это явление называется *лавинным эффектом*. Лавинный эффект повышает криптостойкость системы.

Здесь уместно подчеркнуть, что рассматриваемый алгоритм задуман так, чтобы добиться большего влияния каждого бита открытого текста на шифртекст. В табл. 5.2 показан принцип рассматриваемого расширения.

Таблица 5.2

Расширяющая перестановка

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

Результирующий 48-разрядный код преобразуется далее в 32-разрядный с помощью *S*-матриц. На выходе *S*-матриц осуществляется перестановка символов согласно рекомендуемой схеме перестановок.

Всего используется 8 *S*-матриц (или *S*-блоков). Каждый такой блок имеет 6 входов и 4 выхода, т. е. на его выходе формируется 4-битное слово, значение которого определяется входным 6-битным словом.

В табл. 5.3 показаны все 8 *S*-блоков.

Входное 6-битное слово на входе каждого блока задает адрес ячейки соответствующей матрицы, причем 1-й и 6-й биты задают строку, а остальные – столбец матрицы (адреса начинаются с числа 0).

Выходным 4-битным сообщением данного блока будет бинарное представление содержимого ячейки таблицы с указанными адресами.

Таблица 5.3

S-блоки алгоритма DES

S-БЛОК 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S-БЛОК 2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S-БЛОК 3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S-БЛОК 4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S-БЛОК 5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S-БЛОК 6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S-БЛОК 7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S-БЛОК 8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Пример 5.2. Определить выходное сообщение 8-го *S*-блока, если на его вход подается сообщение 101011. Первый и последний биты (11) определяют третью строку матрицы. Четыре остальных (0101) – адрес столбца. Последнее двоичное число соответствует десятичному числу 5. Это значит, что следует выбрать 6-й слева (первый имеет нулевой – 0000 – адрес) столбец матрицы. На пересечении указанных строки и столбца находится ячейка, содержимым которой является число 10. Таким образом, выходным значением данного блока будет 4-разрядное число 1010.

Подстановка с помощью *S*-блоков является ключевой операцией всего алгоритма. Поскольку она нелинейная (в отличие от остальных), то обеспечивает в наибольшей степени криптостойкость алгоритма DES.

Слово из 32 разрядов на выходе *S*-блоков далее преобразуется в блоке пермутации, или *P*-блоке (на рис. 5.2 – блок перестановки разрядов). Этот процесс перестановки поясняет табл. 5.4.

Таблица 5.4

Перестановка в *P*-блоке

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

Заключительная перестановка (инверсная перестановка на рис. 5.1) обратна по отношению к начальной.

Преобразования ключей. Первоначальный 64-битный ключ сокращается до 56-битного путем отбрасывания битов четности. Остаются 56 битов, последовательность которых показана в табл. 5.5.

Таблица 5.5

Таблица начальной перестановки битов 56-разрядного ключа

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

После этого для каждого из 16 раундов генерируется свой 48-битный подключ. Для этого 56 битов сначала делятся пополам. Затем над каждой половиной выполняется операция сдвига на 1 либо 2 бита (в зависимости от раунда, см. табл. 5.6).

Таблица 5.6

Число битов сдвига ключа в каждом раунде

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

Полученная комбинация ключа будет исходной для следующего раунда. А в текущем раунде выполняется основная операция – сжимающая перестановка: из 56 битов будут выбраны 48 и их последовательность будет изменена. Принцип такого преобразования показан в табл. 5.7.

Таблица 5.7

Сжимающая перестановки ключа

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Как следует из таблицы, например, 33-й бит сдвинутого ключа перемещается на 35-ю позицию, а 18-й бит отбрасывается.

Расшифрование в DES. Осуществляется в обратной последовательности. Компоненты алгоритма подобраны так, чтобы для зашифрования и расшифрования применялся одинаковый алгоритм. Понятно, что ключи используются в обратном порядке в сравнении с зашифрованием.

Криптостойкость DES. Выше было отмечено, что этот важнейший параметр алгоритма определяется не только ключом, но и преобразованием блока открытого текста или шифртекста с помощью *S*-матриц. Что касается ключа и его преобразований, то рассматриваемому алгоритму свойственна так называемая *проблема слабых ключей*. Вспомним, что 56-битный ключ делится пополам. Если все биты каждой половины равны 0 или 1, то во всех раундах будет использоваться одинаковый ключ. Кроме того, существуют пары различных ключей, которые при зашифровании превращают открытые тексты в одинаковый шифртекст.

В 1976 году У. Диффи и М. Хеллман утверждали, что специализированный компьютер стоимостью 20 млн долларов сможет восстановить ключ за 1 день.

В 1990 году Э. Бихам и А. Шамир предложили новый метод криптоанализа – *дифференциальный*, который оказался более эф-

фективным в сравнении с лобовым вскрытием. Криптоаналитик работает с парами шифртекстов, открытые тексты которых имеют некоторые отличия (разности; под разностью здесь понимается операция XOR). Метод предполагает анализ эволюции этой разности в процессе прохождения открытых текстов через раунды алгоритма с использованием одного и того же ключа.

Линейный криптоанализ алгоритма означает следующее. Если выполнить операцию XOR над некоторой частью открытого текста, затем над некоторой частью шифртекста, а затем над полученными результатами (похоже на пример 5.1), то получается некая последовательность, которая представляет собой XOR некоторых битов ключа. Используя фрагменты открытых текстов и соответствующих им шифртектов, можно найти ключ.

Реальность такова, что к началу 1990-х годов рассматриваемый алгоритм был отнесен к числу обеспечивающих низкую криптостойкость.

В настоящее время он все еще находит ограниченное применение.

5.2. Модификации алгоритма DES

Одним из очевидных достоинств алгоритма DES является сравнительно высокая скорость реализации. Это связано, в основном, с малым размером ключа, т. е. преимущество переходит в недостаток и наоборот. Одно из направлений совершенствования алгоритма – использование базового DES в качестве компонента другого алгоритма. Таким другим алгоритмом может быть тот же DES.

Наиболее простой метод объединения или наращивания DES состоит в использовании *двух алгоритмов с различными ключами*:

$$\text{зашифрование: } C = E_{K_2}(E_{K_1}(M)),$$
$$\text{расшифрование: } M = D_{K_1}(D_{K_2}(M)).$$

Криптостойкость такого преобразования значительно возрастает. Если для 56-битного ключа нужно выполнить максимально 2^{56} попыток лобовой атаки и при этом требуется хранить в памяти компьютера 2^{56} 64-битных блоков, или 10^{17} байт, то при двойном DES число попыток взлома возрастает до 2^{56} , а объем требуемой памяти – до 2^{115}, или примерно до $5 \cdot 10^{33}$ байт.

Следующий вариант объединения предложен У. Тачменом. Он предусматривает выполнение формально трех DES (*тройной DES* или 3DES). Один из вариантов такого подхода предполагает следующие операции: отправитель сначала шифрует сообщение первым ключом, зачем расшифровывает результат вторым ключом и, наконец, опять шифрует первым ключом:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(M))),$$

получатель соответственно расшифровывает сообщение первым ключом, зашифровывает вторым, расшифровывает первым:

$$M = D_{K_1}(E_{K_2}(D_{K_1}(C))).$$

В литературе такой режим часто называют *зашифрование-расшифрование-зашифрование* (*Encrypt-Decrypt-Encrypt*, EDE).

Еще большей криптостойкостью обладает 3DES при использовании трех различных ключей:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(M))),$$
$$M = D_{K_1}(E_{K_2}(D_{K_3}(C))).$$

Существуют и иные схемы многократного шифрования.

5.3. Другие блочные алгоритмы

Кратко охарактеризуем некоторые из известных блочных алгоритмов симметричной криптографии.

Алгоритм Lucifer. Предложен У. Тачменом и Х. Файстелом. Алгоритм представляет собой последовательность перестановок и подстановок. Основные блоки напоминают алгоритм DES. Вспомним, что в алгоритме DES результат функции *f* складывается операцией XOR с входом предыдущего раунда, образуя вход следующего раунда (см. рис. 5.1). *S*-блоки алгоритма Lucifer имеют 4-битные входы и выходы. Причем для выбора одного из двух возможных блоков используется бит ключа.

Алгоритм IDEA (International Data Encryption Algorythm – международный алгоритм шифрования данных). Работает с 64-битными блоками данных и 128-битными ключами. Является подстановочно-перестановочным алгоритмом. Блок данных делится на четыре 16-битных блока, являющиеся входами первого раунда.

Весь алгоритм состоит из 8 раундов. В каждом раунде 4 блока подвергаются операциям XOR, сложениям и умножениям друг с другом и шестью 16-битными подключами. Между раундами второй и третий блоки данных меняются местами. В конечной операции четыре 16-битных блока данных объединяются с 16-битными подключами.

Программная реализация IDEA выполняется примерно вдвое быстрее, чем реализация DES.

Алгоритм ГОСТ 28147–89. Разработан в Советском Союзе. Работает с 64-битными блоками данных и 256-битными ключами. Алгоритм состоит из 32 раундов. Как и в DES, блок данных разбивается на левую (L) и правую (R) части. Один раунд алгоритма показан на рис. 5.3.

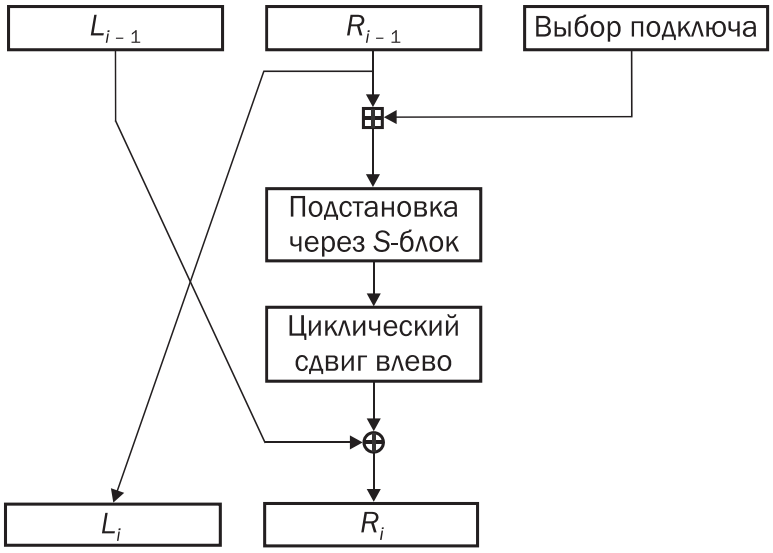


Рис. 5.3. Один раунд алгоритма ГОСТ 28147–89

В i-м раунде выполняются следующие операции:

$$\left.\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} + f(R_{i-1}, K_{i-1}). \end{aligned}\right\} \qquad (5.2)$$

Сравнив выражения (5.1) и (5.2), Вы не найдете отличий. Алгоритм предусматривает, как и в DES, использование 8 S-блоков. 256-битный ключ разбивается на восемь 32-битных подключей, каждый из которых используется в одном раунде. Расшифрование выполняется в обратной последовательности. S-блоки, как и в алгоритме Lucifer, имеют по 4 входа и 4 выхода.

В анализируемом алгоритме отсутствует расширяющая перестановка в сравнении с DES. Это уменьшает *лавинный эффект* и, соответственно, снижает криптостойкость.

Алгоритм Blowfish. Автором является Б. Шнайер. Алгоритм не запатентован и его программная реализация общедоступна. Алгоритм создан для линий связи, где не предусмотрена частая смена ключа. Работает с 64-битными блоками данных. Предусмотрена возможность расширения ключа до 448 битов. Алгоритм состоит обычно из 16 раундов (может быть 12 или 20), в каждом из которых выполняются операции подстановки и перестановки, зависящие от ключа. Используются только сложения и XOR над 32-битными словами. Каждый из 4 *S*-блоков хранит по 256 чисел.

Формально данный алгоритм, как и выше рассмотренные, представляет собой *сеть Фейстела*. Эта сеть описывается соотношениями (5.1) (или (5.2)).

Для сравнительной оценки рассмотренных алгоритмов в табл. 5.8 приведена скорость шифрования (в условных единицах) для каждого из них.

Таблица 5.8

Скорость шифрования (в условных единицах) блочных симметричных шифров

Алгоритм	Скорость шифрования
Blowfish (12 раундов)	182
Blowfish (16 раундов)	135
Blowfish (20 раундов)	110
DES	35
IDEA	70
Lucifer	52
3DES	12

Как видим, самую низкую скорость из рассмотренных дает тройной DES, а Blowfish является наиболее скоростным.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Охарактеризуйте общую схему алгоритма DES.
2. Охарактеризуйте последовательность операций одного раунда алгоритма DES.

3. В чем сущность и польза лавинного эффекта в криптосистемах?

4. Опишите формально сеть Фейстела.

5. Поясните сущность дифференциального и линейного методов криптоанализа.

5.4. Асимметричная криптография

5.4.1. Основы асимметричной криптографии

Две проблемы, связанные с практическим использованием симметричных криптосистем (хранение и обмен ключевой информацией), стали важными побудительными мотивами для разработки принципиально нового класса методов шифрования: криптографии с открытым ключом, или асимметричной криптографии.

Концепция нового подхода предложена Уитфилдом Диффи (Whitfield Diffie) и Мартином Хеллманом (Martin Hellman), а также, независимо, Ральфом Мерклом (Ralph Merkle). В основу положена идея использовать ключи парами: один – для зашифрования (*открытый* или *публичный ключ*), другой – для расшифрования (*тайный ключ*). Таким образом, в отличие от симметричных систем, ключи K_1 и K_2 (см. рис. 3.2) отличаются. Несекретный ключ может передаваться по открытому каналу. Его знание не дает злоумышленнику возможности получить доступ к информации, содержащейся в сообщении.

У. Диффи и М. Хеллман в 1976 году концепцию трансформировали в алгоритм (в литературе соответствующий алгоритм называется *алгоритмом Диффи – Хеллмана*). Начиная с этого события, было создано множество алгоритмов, основанных на концепции открытых ключей. Как правило, эти алгоритмы основываются на решении трудных задач, которые мы рассматривали выше: вычисление дискретного логарифма и разложение чисел на множители. Некоторые из созданных алгоритмов в силу различных причин пригодны для шифрования, другие – для генерации электронной цифровой подписи (ЭЦП). И лишь три из созданных алгоритмов используются в обоих случаях: алгоритмы RSA, Эль-Гамаля и Рабина.

Алгоритм Диффи – Хеллмана для распределения ключей: абоненты A и B могут воспользоваться этим алгоритмом для об-

мена ключевой информацией по открытым каналам. Предварительно стороны выбирают большие простые числа n и g.

Протокол обмена:

1. A выбирает случайное большое число x, вычисляет

$$X = g^x \bmod n$$

и результат вычисления отсылает B.

2. B выбирает случайное большое число y, вычисляет

$$Y = g^y \bmod n$$

и результат вычисления отсылает A.

3. A вычисляет $k_1 = Y^x \bmod n$.

4. B вычисляет $k_2 = X^y \bmod n$.

Таким образом, $k_1 = k_2 = g^{xy} \bmod n = k$.

Это число сторонами может использоваться как совместный ключ (секретный). Для того чтобы третья сторона смогла вычислить значение k, она должна вычислить значение дискретного логарифма.

Протокол Диффи – Хеллмана является уязвимым для атаки, называемой «*человек в середине*»: злоумышленник C может перехватить открытое значение, посылаемое от A к B, и послать вместо него свое открытое значение. Затем он может перехватить открытое значение, посылаемое от B к A, и также передать вместо него свое открытое значение. Тем самым C получит общие секретные ключи с A и B и сможет читать и/или модифицировать сообщения, передаваемые от одной стороны к другой.

Ранцевый алгоритм, разработанный Р. Мерклом и М. Хеллманом, стал первым алгоритмом шифрования с открытым ключом широкого назначения.

Проблема *укладки ранца* формулируется просто. Дано множество предметов различного веса. Спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению? Более формально задача формулируется так: дан набор значений M_1, M_2, …, M_n и суммарное значение S. Требуется вычислить значения b_n такие, что:

$$S = b_1 \cdot M_1 + b_2 \cdot M_2 + \ldots + b_n \cdot M_n.$$

Здесь b_i может быть либо нулем, либо единицей. Значение $b_i = 1$ означает, что предмет M_i кладут в рюкзак, а $b_i = 0$ – не кладут.

Пусть, например, веса предметов имеют значения 1, 5, 6, 11, 14 и 20. При этом можно упаковать рюкзак так, чтобы его вес стал

равен 22, использовав предметы весом 5, 6 и 11. Невозможно упаковать рюкзак так, чтобы его вес стал равен 24. Время, необходимое для решения этой проблемы, в общем случае возрастает как экспоненциальная функция от количества предметов.

В основе алгоритма, предложенного Мерклом и Хеллманом, лежит идея шифрования сообщения на основе решения серии задач укладки ранца. Предметы из «кучи» выбираются с помощью блока открытого текста, длина которого (в битах) равна количеству предметов в куче. При этом биты открытого текста соответствуют значениям b_i, а шифртекст является полученным суммарным весом. Пример текста, зашифрованного с помощью задачи укладки ранца, показан в табл. 5.9.

Существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая, как принято считать, нет. Легкий для укладки ранец можно превратить в трудный. В качестве открытого ключа можно применить трудный для укладки ранец, который легко использовать для шифрования, но невозможно – для расшифрования сообщений. А в качестве закрытого ключа применяют легко разрешимый для укладки ранец, который предоставляет простой способ расшифрования сообщения. Тому, кто не знает закрытый ключ, придется попытаться решить трудную задачу укладки ранца.

Таблица 5.9

Шифрование на основе укладки ранца

Открытый текст	111001	010110	000000	011000
Ранец	1 5 6 11 14 20			
Вычисление шифра	1 + 5 + 6 + 20 =	5 + 11 + 14 =	0	5 + 6 =
Шифртекст	32	30	0	11

5.4.2. Алгоритм RSA

Вскоре после появления алгоритма укладки ранца Меркла был создан первый полноценный алгоритм с открытым ключом, который можно использовать и для шифрования, и для создания цифровых подписей – алгоритм RSA. Этот алгоритм также является самым популярным. Названный в честь трех изобретателей – Рона Ривеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Адлемана (Leonard Adleman) – этот алгоритм многие годы противостоит многочисленным попыткам криптоаналитического вскрытия.

Безопасность алгоритма основана на трудоемкости разложения на множители (факторизации) больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел разрядностью 100–200 десятичных цифр или даже больше. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу равносильно разложению числа на два больших простых множителя.

Описание алгоритма RSA. Предварительный этап – *генерация ключа*, состоящего из трех чисел.

1. Случайным образом выбираются два больших простых числа p и q. Рассчитывается произведение $n = p \cdot q$. Это первое число, входящее в ключ.

2. Вычисляется функция Эйлера (см. выражение (4.3) на с. 52):

$$\varphi(n) = (p - 1)(q - 1).$$

3. Случайным образом выбирается простое число e – вторая часть ключа, которое удовлетворяет условиям: $e < \varphi(n)$; e и $\varphi(n)$ должны быть взаимно простыми числами.

4. Вычисляется число d – третья часть ключа, которое является обратным числу e (см. выражения (4.4) и (4.5) на с. 54), т. е.

$$e \cdot d \equiv 1 \bmod \varphi(n).$$

Пара чисел (e, n) делается открытым ключом и помещается в общедоступный справочник (база данных), а о числах p, q можно забыть (но это также является тайной информацией). Пара (d, n) – секретный ключ (понятно, что секретным является лишь значение первого числа из этой пары); эти числа также являются взаимно простыми. Первая обычно используется для зашифрования, другая – для расшифрования. Не вдаваясь в детали, отметим важную особенность генерации и использования ключа: числа e и d можно поменять местами. Только в этом случае первое из них будет тайным (входит в ключ для расшифрования), второе – открытым.

Следующий этап – использование ключа.

Зашифрование. Если шифруется сообщение M, состоящее из r блоков: $m_1, m_2, \ldots, m_i, \ldots, m_r$, то шифртекст C будет состоять из такого же числа (r) блоков, представляемых числами:

$$c_i = (m_i)^e \bmod n. \qquad (5.3)$$

Расшифрование. Для расшифрования каждого зашифрованного блока производится вычисление вида

$$m_i = (c_i)^d \bmod n. \qquad (5.4)$$

Рассмотрим примеры.

Пример 5.3. Генерация ключа. Принимаем $p = 11$, $q = 5$.

Вычисляем $n = 11 \cdot 5 = 55$.

Определяем функцию Эйлера: $\varphi(55) = (11 - 1)(5 - 1) = 40$.

Выбираем ключ зашифровывания $e = 7$, который удовлетворяет условиям $7 < 40$; НОД $(7, 40) = 1$.

Определяем d – ключ расшифровывания – из уравнения

$$7 \cdot d \equiv 1 \bmod 40.$$

Рассмотрим способ нахождения d.

Для решения уравнения $7 \cdot d \equiv 1 \bmod 40$ используем алгоритм Евклида:

$$40 = 7 \cdot 5 + 5;$$
$$7 = 5 \cdot 1 + 2;$$
$$5 = 2 \cdot 2 + 1;$$
$$2 = 1 \cdot 2 + 0.$$

Обратная подстановка дает:

$$1 = 5 - 2 \cdot 2 = 5 - (7 - 5 \cdot 1)2 = 5 \cdot 3 + 7(-2) =$$
$$= (40 - 7 \cdot 5)3 + 7(-2) = 40 \cdot 3 + 7(-17).$$

Поскольку $-17 = 23 \bmod 40$, то $d = 23$.

Зашифровываем сообщение $M = 15$ (сообщение состоит из одного блока), используя выражение (5.3):

$$C = 15^7 \bmod 55 = 5.$$

Для расшифрования C воспользуемся формулой (5.4):

$$M = 5^{23} \bmod 55 = 15.$$

Пример 5.4. Пусть $p = 47$ и $q = 71$, тогда $n = 3337$.

Ключ e не должен иметь общих множителей с $(p - 1)(q - 1) = 46 \cdot 70 = 3220$; принимаем $e = 79$, тогда

$$d = 79^{-1} \bmod 3220 = 1019.$$

Пусть сообщение имеет следующий вид: $M = 688232687$ и длина блока равна 3:

$$m_1 = 688,\ m_2 = 232,\ m_3 = 687.$$

Зашифрование: $c_1 = 688^{79} \bmod 3337 = 1570$ и т. д.

Расшифрование: $m_1 = 1570^{1019} \bmod 3337 = 688$ и т. д.

Пример 5.5. Необходимо зашифровать/расшифровать сообщение M = ВСА при $p = 11$ и $q = 3$.

Вычисляем элементы ключа: $n = 33$; $\varphi(n) = 10 \cdot 2 = 20$; $e = 7$ (взаимно простое с 20) и $d = 3$.

Сообщение представляем в числовой форме: $m_1 = 2$, $m_2 = 3$, $m_3 = 1$ (номера соответствующих букв в алфавите).

Зашифрование: $c_1 = 2^7 \bmod 33 = 29$; $c_2 = 3^7 \bmod 33 = 9$; $c_3 = 1^7 \bmod 33 = 1$.

Поскольку вычисления ведутся по модулю 33, целесообразно предположить, что открытые сообщения строятся на основе алфавита такой же мощности (33). Предположим также, что нулевым символом этого алфавита будет число «0», а символами с 1 по 26 – латинский алфавит, с 27 по 32 – соответственно «1»–«6». В соответствии с этим шифртекст сообщения можно представить в виде чисел ($C = 290901$) либо в виде символов условного алфавита: C = 3IA.

Расшифрование: $m_1 = 29^3 \bmod 33 = 2$ (соответствует букве «В»); $m_2 = 9^3 \bmod 33 = 3$ («С»); $m_3 = 1^3 \bmod 33 = 1$ («А»).

Криптостойкость алгоритма. Зависит от трудоемкости решения проблемы разложения на множители больших чисел. «Лобовой метод» вскрытия системы RSA заключается в нахождении числа d, обратного e по модулю $\varphi(n)$. Это легко сделать, если известны числа p и q. Отметим, что математически не доказано, что для восстановления сообщения по шифртексту и по значению открытого ключа нужно разложить n на множители. Тем не менее именно этот подход является наиболее очевидным. Известно[15], что факторизованы числа длиной более 512 битов. Значит, нужно выбирать n больше этого значения. Кроме того, числа p и q не должны быть слишком близкими друг к другу.

Существует одна важная особенность. Если группа пользователей применяет одно и то же значение модулю, но каждый имеет разный ключ, и одно и то же сообщение шифровалось разными пользователями, то вероятность взлома шифра значительно возрастает.

---

[15] О стойкости // Официальный сайт НИИ прикладных проблем математики и информатики БГУ [Электронный ресурс]. 2014. URL: http://apmi.bsu.by/ (дата обращения: 20.08.2015).

Существует и другое направление взлома шифра. Криптоаналитик располагает шифртекстом $C = E_e(M)$. Цель – восстановить M (расшифровать C). Для этого он шифрует известным открытым ключом произвольное сообщение M'. Получает C'. Если $C = C'$, то получено M ($M = M'$), в ином случае шифруется другое сообщение M'' и т. д.

В заключение отметим, что аппаратно реализованный алгоритм RSA работает более чем в 1000 медленнее, чем алгоритм DES. При программной же реализации обоих алгоритмов быстродействие первого из них хуже примерно в 100 раз.

5.4.3. Алгоритм Эль-Гамаля

Данный алгоритм является альтернативой алгоритму RSA и, при равном значении ключа, обеспечивает ту же криптостойкость. Стойкость алгоритма Эль-Гамаля основана на трудности вычисления дискретных логарифмов.

Предварительный этап – *генерация ключа*, состоящего из четырех чисел:

1. Выбирается простое число p и два случайных числа, меньших чем p: числа x и g.

2. Далее вычисляется

$$y = g^x \bmod p.$$

Открытый ключ: y, g и p; тайный ключ: x.

Следующий этап – *использование ключа*.

В отличие от алгоритма RSA, рассматриваемый алгоритм предусматривает использование дополнительного параметра, который обозначим k. Использование этого параметра снижает вероятность взлома шифра. Это число является секретным и должно быть взаимно простым с $p - 1$.

Зашифрование. Если шифруется сообщение M, состоящее из r блоков: $m_1, m_2, \ldots, m_i, \ldots, m_r$, то шифртекст C будет состоять из такого же числа (r) блоков, представляемых *парой чисел*:

$$a_i = g^k \bmod p, \tag{5.5}$$

$$b_i = (y^k \cdot m_i) \bmod p. \tag{5.6}$$

Расшифрование. Для расшифрования производится вычисление вида:

$$m_i = b_i/(a_i)^x \bmod p. \tag{5.7}$$

Так как $(a_i)^x$ в силу формулы (5.5) можно заменить на $g^{kx} \bmod p$ и учитывая то, что $y = g^x \bmod p$, можем получить подтверждение справедливости выражения (5.7):

$$b_i/(a_i)^x \equiv y^k \cdot m_i/(a_i)^x \bmod p \equiv g^{kx} \cdot m_i/g^{kx} \bmod p = m_i \bmod p.$$

Справедливо также

$$m_i = b_i \cdot (a_i)^{p-1-x} \bmod p. \qquad (5.8)$$

Рассмотрим пример.

Пример 5.6. Необходимо выполнить прямую и обратную криптографическую процедуру, если M = Абрамов.

Выбираем $p = 23$, $g = 5$ и $x = 3$. Вычисляем $y = g^x \bmod p = 5^3 \bmod 23 = 10$.

Предполагаем, что исходный текст строится на основе алфавита мощностью p. В этом случае представление сообщения M в виде цифровой последовательности будет выглядеть так: 01 02 18 01 14 16 03, т. е. $m_1 = 1$, $m_i = 2$, $m_3 = 18$ и т. д.

Выбираем $k = 7$ (для шифрования каждого символа следует выбирать разные k – *принцип вероятностного шифра*). В нашем примере для упрощения примем k неизменным при шифрования каждого символа текста. В этом случае параметр a_i будет одинаковым для всех значений i:

$$a_i = g^k \bmod p = 57 \bmod 23 = 17.$$

Используем формулу (5.6) для вычисления второй части шифртекста:

для первой буквы (А): $b_1 = (y^{k_1} \cdot M_1) \bmod p = (10^7 \cdot 1) \bmod 23 = 14$,

для второй буквы (б): $b_2 = (y^{k_2} \cdot M_2) \bmod p = (10^7 \cdot 2) \bmod 23 = 05$,

для третьей буквы (р): $b_3 = (y^{k_3} \cdot M_3) \bmod p = (10^7 \cdot 18) \bmod 23 = 22$

и т. д.

Шифртекст (пары a_i и b_i): C = 1714 1705 1722.

Как видим, длина зашифрованного сообщения действительно удвоилась.

Расшифрование. Используем выражение (5.7): $m_1 = (b_1 ((a_1)^x)^{-1}) \bmod p = (14 ((17)^3)^{-1}) \bmod 23 = (14 \cdot (4913)^{-1}) \bmod 23 = ((14 \bmod 23) \times \times ((4913)^{-1} \bmod 23)) \bmod 23 = (14 \cdot 5) \bmod 23 = 70 \bmod 23 = 1$, т. е. m_1 = А.

При этом мы приняли во внимание: $(4913)^{-1} \bmod 23 = 5$, так как $4913 \cdot 5 \bmod 23 = 1$.

Или используем формулу (5.8):

$$m_1 = (b_1 (a_1)^{p-1-x}) \bmod p = 14 \cdot 17^{23-1-3} \bmod 23 = 14 \cdot 17^{19} \bmod 23 =$$
$$= (14 \cdot 2390724356851513248471 53) \bmod 23 = 1.$$

Аналогично получаем остальные значения зашифрованного сообщения.

Если абоненты A и B по согласованию выбрали одинаковые значения p и g, то для обмена зашифрованными сообщениями они могут воспользоваться идеей, реализованной в вышерассмотренном алгоритме Диффи – Хеллмана (в литературе иногда указывается, что алгоритм Эль-Гамаля подпадает под действие патента, полученного У. Диффи и М. Хеллманом). Сторона A генерирует секретный ключ $x_A < p$ и вычисляет открытый ключ:

$$y_A = (g^{x_A}) \bmod p, \tag{5.9}$$

сторона B выбирает число $x_B < p$ и с его помощью зашифровывает передаваемое сообщение M следующим образом

$$y_B = (g^{x_B}) \bmod p, \tag{5.10}$$
$$C = M \oplus (y_A)^{x_B} \bmod p. \tag{5.11}$$

В последнем случае суммирование ведется по модулю два, а суммируемые числа представляются в двоичном виде.

Предварительно стороны обмениваются открытыми ключами y_A и y_B. Понятно, что закрытые ключи (x_A и x_B) каждая сторона хранит в тайне.

Сторона A, получив сообщение в виде y_B и C, восстанавливает его:

$$M = ((y_B)^{x_A} \bmod p) \oplus C. \tag{5.12}$$

Пример 5.7. Пусть $p = 11$; $q = 3$; $x_A = 7$; $x_B = 4$; $M = 6$.

Открытый ключ, посылаемый стороне B, равен (см. выражение (5.9)):

$$y_A = 3^7 \bmod 11 = 2187 \bmod 11 = 9.$$

Сообщение, зашифрованное на стороне B, имеет вид (5.10):

$$y_B = 3^4 \bmod 11 = 81 \bmod 11 = 4;$$

для вычисления C предварительно определим второе слагаемое в выражении (5.11):

$$(y_A)^{x_B} \bmod p = 9^4 \bmod 11 = 5 \text{ (в двоичном виде – 101).}$$

И наконец, $C = 110 \oplus 101 = 011$, где двоичным представлением M будет последовательность 101.

Сторона A, получив зашифрованное сообщение в виде y_B и C, расшифровывает его, используя выражение (5.12): $4^7 \bmod 11 = 5$ (101) и далее: $101 \oplus 011 = 110$ (десятичное число 6, т. е. $M = 6$).

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. На чем основана идея обмена ключевой информацией Диффи и Хеллмана?

2. Сформулируйте задачу укладки ранца и покажите примеры ее решения.

3. Зашифруйте и расшифруйте сообщения (свои фамилию, имя, отчество) с использованием алгоритма Эль-Гамаля для ключевых значений, представленных следующей таблицей.

Вариант	p	g	x_A	x_B	Вариант	p	g	x_A	x_B
1	23	11	14	21	16	23	14	17	16
2	29	11	14	21	17	29	14	17	16
3	31	11	14	21	18	31	14	17	16
4	37	11	14	21	19	37	14	17	16
5	43	11	14	21	20	43	14	17	16
6	23	12	15	19	21	23	15	18	13
7	29	12	15	19	22	29	15	18	13
8	31	12	15	19	23	31	15	18	13
9	37	12	15	19	24	37	15	18	13
10	43	12	15	19	25	43	15	18	13
11	23	13	16	18	26	23	16	21	12
12	29	13	16	18	27	29	16	21	12
13	31	13	16	18	28	31	16	21	12
14	37	13	16	18	29	37	16	21	12
15	43	13	16	18	30	43	16	21	12

5.4.4. Криптосистемы на эллиптических кривых

В 1985 году Нил Коблиц (Neal Koblitz) и (независимо) Виктор Миллер (Victor Miller) предложили использовать эллиптические кривые для создания криптосистем с открытым ключом.

Их безопасность, как правило, основана на трудности решения задачи дискретного логарифмирования в группе точек эллип-

тической кривой над *конечным полем*. Этим и обусловлена их высокая криптостойкость по сравнению с другими алгоритмами.

Эллиптические кривые – математический объект, который может быть определен над любым полем. В криптографии обычно используются конечные поля. Для точек на эллиптической кривой вводится операция сложения, которая играет ту же роль, что и операция умножения в криптосистемах RSA и Эль-Гамаля. Еще одним преимуществом криптосистем на эллиптических кривых является высокая скорость обработки информации. Например, уровень стойкости, который достигается, скажем, в RSA при использовании 1024-битных ключей, в системах на эллиптических кривых реализуется при том же параметре длиной 160 битов.

Криптосистемы на эллиптических кривых, как, впрочем, и другие криптосистемы с открытым ключом, нецелесообразно применять для шифрования больших объемов данных. Но зато их можно эффективно использовать для систем цифровой подписи и ключевого обмена. С 1998 года использование эллиптических кривых для решения криптографических задач, таких как цифровая подпись, было закреплено в стандартах США ANSI X9.62 и FIPS 186–2, а в 2001 году аналогичный стандарт был принят в России – ГОСТ Р 34.10–2001.

5.4.4.1. Представление и описание эллиптической кривой

Для описания эллиптической кривой используется *алгебраическая геометрия*[16].

Эллиптическая кривая (ЭК, EC – Elliptic Curve) – это не эллипс. Она называется так потому, что описывается кубическим уравнением, подобным тем, которые используются для вычисления кривой эллипса.

В общем случае кубические уравнения для эллиптических кривых имеют вид

$$y^2 + a \cdot x \cdot y + b \cdot y = x^3 + c \cdot x^2 + d \cdot x + e, \qquad (5.13)$$

где a, b, c, d, и e являются действительными числами, удовлетворяющими некоторым простым условиям. Такие уравнения называются еще *уравнениями третьего порядка*, поскольку в них наивысший показатель степени равен трем.

[16] В некотором приближении алгебраическая геометрия занимается алгебраическими множествами и алгебраическими отображениями.

С другой стороны, эллиптическая кривая – это набор точек (x, y), удовлетворяющих уравнению (5.13) для переменных (x, y) и констант (a, b, c, d, e), принадлежащих множеству F, где F – поле.

Вспомним некоторые важные характеристики поля.

Конечным полем называется алгебраическая система, которая состоит из конечного множества F и двух бинарных операций (сложения и умножения).

Порядком поля называется количество элементов в поле (во множестве F).

Фундаментальным является следующее условие: конечное поле порядка p существует тогда и только тогда, когда является простым. **Такое поле обозначается GF(p)**.

В криптографии широкое распространение получили поля, где p – простое число, и поля характеристики 2: GF(2^m). Уравнение эллиптической кривой над полем первого типа можно привести к виду

$$y^2 = (x^3 + a \cdot x + b) \bmod p, \tag{5.14}$$

при этом константы должны удовлетворять условию

$$4 \cdot a^3 + 27 \cdot b^2 \neq 0 \pmod p.$$

Пример 5.8. Рассмотрим эллиптическую кривую, описываемую уравнением $y^2 = x^3 + 3 \cdot x + 2$ над полем GF(5). На этой кривой, в частности, лежит точка $(x = 1, y = 1)$, так как

$$1^2 = (1^3 + 3 \cdot 1 + 2) \bmod 5.$$

Поля характеристики 2, GF(2^m), называют также *бинарными конечными полями*. У эллиптических кривых над полем GF(2^m) есть одно важное преимущество: элементы поля GF(2^m) могут быть легко представлены в виде n-битных кодовых слов, это позволяет увеличить скорость аппаратной реализации эллиптических алгоритмов.

Над полем GF(2^m) рассматривают два вида эллиптических кривых:

- суперсингулярная кривая:

$$y^2 + a \cdot y = x^3 + b \cdot x + c, \tag{5.15}$$

- несуперсингулярная кривая:

$$y^2 + a \cdot y \cdot x = x^3 + b \cdot x^2 + c. \tag{5.16}$$

Определение эллиптической кривой включает также некоторый элемент, называемый *несобственным элементом* (или *бесконечным элементом*, или *нулевым элементом*).

Таким образом, помимо точек, принадлежащих кривой (координаты которых удовлетворяют уравнениям (5.13)–(5.16)), вводится точка, не лежащая на кривой; если со школьного курса Вы припоминаете что-то про «точку на бесконечности», то это она и есть; будем обозначать ее буквой «*O*».

Например, формальная запись $E_{23}(1, 1)$ будет означать, что мы используем уравнение кривой вида

$$y^2 = (x^3 + x + 1) \bmod 23.$$

Эллиптическая кривая или множество $E_p(a, b)$ состоит из всех точек (x, y), $x > 0$, $y < p$, удовлетворяющих уравнению (5.14), и точки в бесконечности O.

Количество точек в $E_p(a, b)$ будем обозначать $\#E_p(a, b) = N$.

Определение 5.1. ***Порядок эллиптической кривой*** **– это число, которое показывает количество точек кривой над конечным полем.**

Эта величина имеет большое значение для криптографических приложений эллиптических кривых. Чтобы использовать эллиптическую кривую в криптографии, необходимо знать ее порядок. Это обуславливается тем, что от порядка кривой зависит криптостойкость системы на основе соответствующей кривой (уравнения кривой).

Арифметические операции в эллиптической криптографии **производятся над точками кривой.**

Основной операцией является сложение. *Сложение двух точек* легко представить графически (рис. 5.4).

Как видно из рис. 5.4, для сложения точек P и Q необходимо провести между ними прямую линию, которая обязательно пересечет кривую в какой-либо третьей точке ($-R$; иногда эту точку обозначают R). Отразим точку $-R$ относительно горизонтальной оси координат и получим искомую точку: $P + Q$ – результат сложения: $P + Q = R$, т. е. точки R и $-R$ симметричны относительно горизонтальной оси.

По определению, эллиптическая кривая обладает следующим свойством: ***если три ее точки лежат на одной прямой, то их сумма равна O.***

Это свойство позволяет описать основные правила сложения, а также умножения точек эллиптической кривой:

• пусть P и Q – две различные точки эллиптической кривой (рис. 5.4), и P не равно Q. Проведем через P и Q прямую. Она пересечет эллиптическую кривую только в одной точке, называемой $-R$. Точка $-R$ отображается относительно оси X в точку R, равную сумме точек P и Q; **закон сложения точек эллиптической кривой**: $\boldsymbol{P + Q = R}$;

• прямая, проходящая через точки R и $-R$, является вертикальной прямой, которая не пересекает эллиптическую кривую ни в какой третьей точке; если $R = (x, -y)$, то $R + (x, y) = O$. Точка (x, y) является отрицательным значением точки R и обозначается $-R$; таким образом, **по определению** $\boldsymbol{R + (-R) = O}$;

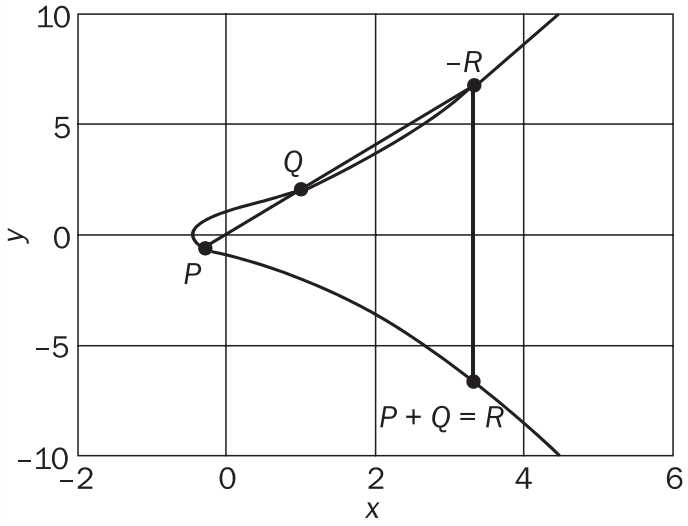


Рис. 5.4. Пояснение к операции сложения двух точек эллиптической кривой $y^2 = x^3 + 2 \cdot x + 1$

• если O – нулевой элемент, то справедливо равенство $O = -O$, а **для любой точки** $\boldsymbol{P}$ **эллиптической кривой имеем** $\boldsymbol{P + O = P}$;

• чтобы сложить точку P с ней самой, нужно провести касательную к кривой в точке P; **закон удвоения точки** $\boldsymbol{P}$**:** $\boldsymbol{P + P = 2 \cdot P}$;

• **умножение точки** $\boldsymbol{P}$ **на целое положительное число** $\boldsymbol{k}$ определяется как сумма k точек P: $\boldsymbol{k \cdot P = P + P + P + \ldots + P}$;

• скалярное умножение осуществляется посредством нескольких комбинаций сложения и удвоения точек эллиптической кривой. Например, точка $25 \cdot P$ может быть представлена как

$$25 \cdot P = (2 \cdot (2 \cdot (2 \cdot (2 \cdot P)))) + (2 \cdot (2 \cdot (2 \cdot P))) + P.$$

На рис. 5.5 приведены примеры некоторых арифметических операций над тремя точками кривой.

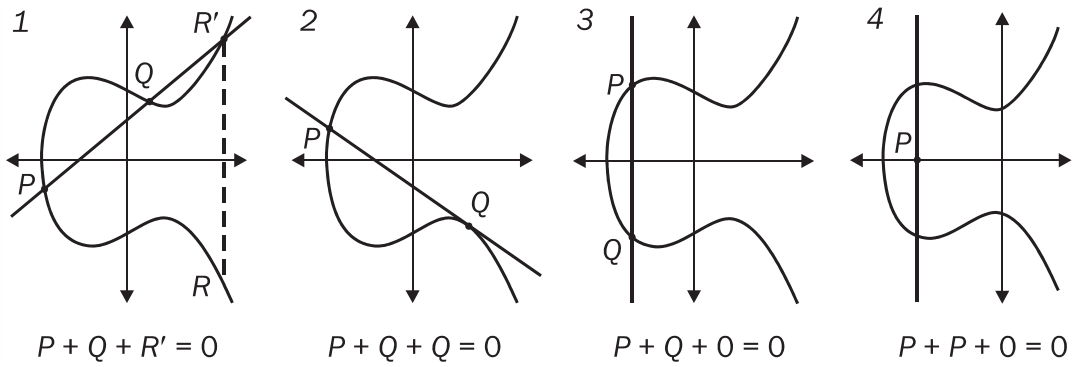


Рис. 5.5. Примеры некоторых арифметических операций над точками эллиптической кривой

Как следует из примеров на данном рисунке, если P и Q – две точки на кривой, то мы можем единственным образом описать третью точку – точку пересечения данной кривой с прямой, проведенной через P и Q (первый пример на рис. 5.5); если прямая, которая используется для выполнения операции сложения точек, является касательной к кривой в точке, то такая точка считается дважды (второй пример); если же прямая параллельна оси ординат, третьей точкой будет точка в бесконечности (третья и четвертая иллюстрации).

Идея, надежность и криптостойкость эллиптической криптографии напрямую связаны с операцией умножения точки на целое число: **задача вычисления дискретного логарифма на эллиптической кривой, заключающаяся в отыскании целого числа x по известным точкам P и $Q = x \cdot P$, является трудноразрешимой.**

Если $P = (x_1, y_1)$ и $Q = (x_2, y_2)$, то $P + Q = (x_3, y_3)$ определяется в соответствии с правилам:

$$x_3 = (\lambda^2 - x_1 - x_2) \bmod p;$$
$$y_3 = (\lambda \cdot (x_1 - x_3) - y_1) \bmod p, \qquad (5.17)$$

где $\lambda = (y_2 - y_1)/(x_2 - x_1)$, если $P \neq Q$, и $\lambda = (3x_1^2 + a)/2y_1$, если $P = Q$.

Из этого следует, что число λ – угловой коэффициент секущей, проведенной через точки $P = (x_1, y_1)$ и $Q = (x_2, y_2)$. При $P = Q$ секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

Пример 5.9. В случае $E_{23}(1, 1)$ – соответствует вышеприведенному уравнению $y^2 = (x^3 + x + 1) \bmod 23$ – для $P = (13, 7)$ имеем $-P = (13, -7)$. Но $-7 \bmod 23 = 16$, таким образом, $-P = (13, 16)$, т. е. $x = 13$, $y = 16$.

Здесь *следует вспомнить, что значение некоторого целого отрицательного числа* $(-k)$ *по модулю* (p) *вычисляется следующим образом*:

$$(-k) \bmod p = -(k \bmod p) + p. \qquad (5.18)$$

Возьмем более общий пример.

Пример 5.10. Рассмотрим эллиптическую кривую E (рис. 5.6), соответствующую уравнению

$$y^2 + y = x^3 - x^2. \qquad (5.19)$$

На этой кривой лежат только четыре точки, координаты которых являются целыми числами. Это точки $A(0, 0)$, $B(1, -1)$, $C(1, 0)$, $D(0, -1)$. На плоскости существует бесконечно удаленная точка $O \in E$, в которой сходятся все вертикальные прямые.

Выполним операцию сложения точек $P, Q \in E$ (рис. 5.7). Для этого проведем прямую линию через точки P и Q, найдем третью точку $-R$ пересечения этой прямой с кривой E. Далее проведем через точку $-R$ вертикальную прямую до пересечения с кривой E в точке R, которая будет искомой суммой: $P + Q = R$.

Применив известные правила к группе точек $\{A, B, C, D, O\}$, получим (см. рис. 5.8):

$$A + A = B; \quad A + B = C; \quad A + C = D; \quad A + D = O,$$

или

$$2 \cdot A = B; \quad 3 \cdot A = C; \quad 4 \cdot A = D; \quad 5 \cdot A = O; \quad 6 \cdot A = A.$$

Еще раз отметим, что важнейшим параметром эллиптической кривой, определяющим возможность ее использования в криптографии, является *количество точек* этой кривой: $\#E_p(a, b) = N$.

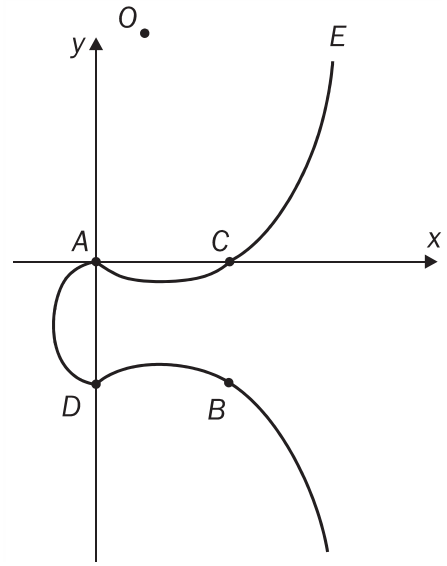


Рис. 5.6. Группа из пяти точек эллиптической кривой *E*; *O* – бесконечно удаленная точка

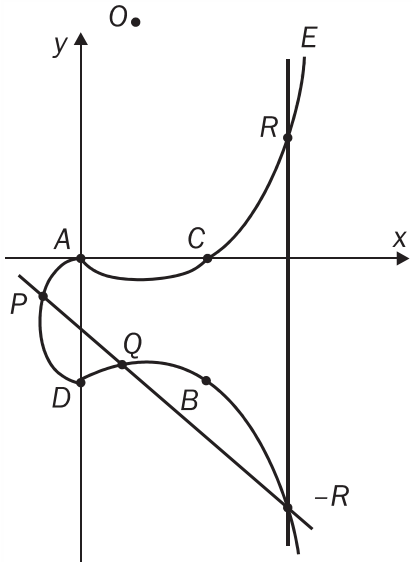
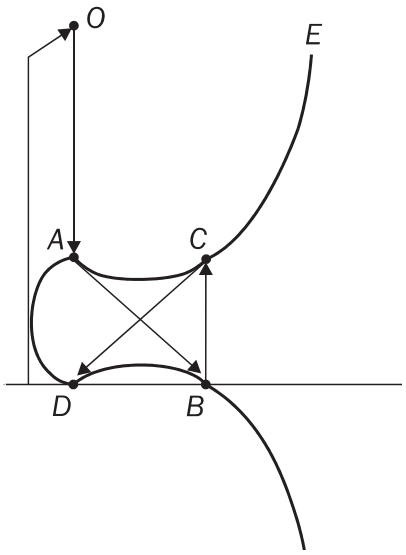


Рис. 5.7. Графическое представление сложения точек: $P + Q = R$



б

Рис. 5.8. Абелева группа ({*A*, *B*, *C*, *D*, *O*}) на кривой *E*

После достаточно подробного рассмотрения операций над точками эллиптических кривых мы увидели определенные правила, по которым выполняются эти операции. Математики говорят, что такие правила, в общем случае, относятся к понятию «*аддитивная абелева группа*» или «*коммутативная абелева группа*».

Определение 5.2. ***Аддитивной абелевой группой*** **называется множество A с операцией сложения, обладающей следующими свойствами:**

1) $a + b = b + a$, $\forall\ a, b \in A$ **(коммутативность);**

2) $(a + b) + c = a + (b + c)$, $\forall\ a, b \in A$ **(ассоциативность);**

3) во множестве A существует такой элемент O (нуль), что $a + O = a$, $\forall\ a \in A$;

4) для любого элемента $a \in A$ **существует такой элемент** $-a \in A$ **(противоположный элемент), что** $a + (-a) = O$.

Нетрудно понять, что если вместо a и b подставить точки кривой E (все точки которой вместе с точкой O и образуют множество A), то мы получим подтверждение корректности всем рассмотренным нами примерам.

Пример 5.11. Рассмотрим кривую $E_7(2, 6)$: $y^2 = x^3 + 2 \cdot x + 6 \pmod 7$, т. е. коэффициентами уравнения являются: $a = 2$ и $b = 6$. Проверим выполнение условия несингулярности кривой (см. выражение (5.14)):

$$(4 \cdot 2^3 + 27 \cdot 6^2) \bmod 7 = 3 \bmod 7 \neq 0.$$

Итак, данная кривая несингулярна. Найдем какую-нибудь (случайную) точку на $E_7(2, 6)$. Пусть $x = 5$, тогда

$$y^2 = (5^3 + 2 \cdot 5 + 6) \bmod 7 = (125 + 10 + 6) \bmod 7 = 1 \bmod 7$$

и $y = 1 \bmod 7$ или (с учетом выражения (5.18)): $y = -1 \bmod 7 = 6 \bmod 7$, поскольку $(-1)^2 = 1$.

Найдены сразу две точки: (5, 1) и (5, 6).

Для поиска новых точек проведем следующие вычисления, используя формулу (5.17). Очередную точку найдем путем вычисления композиции. Вначале найдем $2 \cdot (5, 1)$: это соответствует случаю $P = Q$. Для этого случая

$$\lambda = ((3 \cdot x_1^2 + a) / 2 \cdot y_1) \bmod 7 = ((3 \cdot 5^2 + 2)/ 2 \cdot 1) \bmod 7 = 0 \bmod 7$$

и далее:

$$x_3 = (0 - 2 \cdot 5) \bmod 7 = 4 \bmod 7;$$

$$y_3 = (0\ (5 - 4) - 1) \bmod 7 = 6 \bmod 7.$$

Получили $2 \cdot (5, 1) = (4, 6)$. Можно убедиться, что полученная точка лежит на кривой, подставив ее координаты в уравнение

$$y^2 = (x^3 + 2 \cdot x + 6) \bmod 7.$$

Найдем еще одну точку $3 \cdot (5, 1) = (5, 1) + (4, 6)$:

$$\lambda = (6 - 1)(4 - 5) \pmod 7 = -5 \bmod 7 = 2 \bmod 7;$$
$$x_3 = (22 - 5 - 4) \bmod 7 = 2 \bmod 7;$$
$$y_3 = (2\,(5 - 2) - 1) \bmod 7 = 5 \bmod 7.$$

Итак, найдены четыре точки. Еще раз подчеркнем: для криптографического использования кривой важно знать, сколько всего точек в множестве $E_7(2, 6)$.

Пример 5.12. Пусть $p = 23$. Рассмотрим эллиптическую кривую E:

$$y^2 = (x^3 + x + 1) \bmod 23,$$

т. е. $E_{23}(1, 1)$.

Кривая состоит из следующих точек: (0, 1); (0, 22); (1, 7); (1, 16); (3, 10); (3, 13); (4, 0); (5, 4); (5, 19); (6, 4); (6, 19); (7, 11); (7, 12); (9, 7); (9, 16); (11, 3); (11, 20); (12, 4); (12, 19); (13, 7); (13, 16); (17, 3); (17, 20); (18, 3); (18, 20); (19, 5); (19, 18), т. е $N = 27$.

Пусть $P = (3, 10)$ и $Q = (9, 7)$. Найдем $P + Q$ и $2 \cdot P$.

Пусть $P + Q = (x_3, y_3)$, тогда при

$$\lambda = ((7 - 9)/(9 - 3)) \bmod 23 = 11 \bmod 23$$

имеем:

$$x_3 = (121 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17 \bmod 23;$$
$$y_3 = (11(3 + 6) - 10) \bmod 23 = 89 \bmod 23 = 20 \bmod 23.$$

Таким образом, $P + Q = (17, 20)$.

Найдем теперь точку $2 \cdot P = P + P = (x_3, y_3)$ – с формальной точки зрения это также будет третья точка. Для этого случая

$$\lambda = ((3 \cdot 9 + 1)/9) \bmod 23 = 6 \bmod 23$$

и с учетом выражения (5.18) для вычисления координаты y_3:

$$x_3 = 36 - 6 = 30 \bmod 23 = 7 \bmod 23;$$
$$y_3 = (6 \cdot (3 - 7) - 10) \bmod 23 = (-34) \bmod 23 = 12 \bmod 23.$$

Таким образом, $2 \cdot P = (7, 12)$.

При последовательном выполнении сложения $\boldsymbol{k \cdot P = P + P + P + \ldots + P}$ на каждом шаге будет получаться точка, которая также должна принадлежать $E_p(a, b)$. В силу того что эллиптическая группа содержит конечное множество точек, наступит такой момент, что для некоторых результатов вычислений будет выполняться равенство $\boldsymbol{q \cdot P = O}$ (см. пример 5.10, где $5 \cdot A = O$, т. е. здесь $q = 5$).

Определение 5.3. Наименьшее значение числа *q*, для которого выполняется равенство $q \cdot P = O$, называется *порядком точки P*.

Определение 5.4. Порядок группы точек эллиптической кривой равен числу различных точек ЭК, включая точку *O*.

Для эллиптической кривой E_p (a, b) порядок m группы точек должен удовлетворять неравенству:

$$p + 1 - 2 \cdot p^{1/2} \le m \le p + 1 + 2 \cdot p^{1/2}. \qquad (5.20)$$

В табл. 5.10 приведены результаты последовательного сложения для некоторых точек кривой вида $y^2 = x^3 + 1$.

Таблица 5.10

Результаты выполнения операции *P* + ... + *P* для кривой $E_5(0, 1)$

+	(0, 1)	(0, 4)	(2, 2)	(2, 3)	(4, 0)
P	(0, 1)	(0, 4)	(2, 2)	(2, 3)	(4, 0)
2P	(0, 4)	(0, 1)	(0, 4)	(0, 1)	O
3P	O	O	(4, 0)	(4, 0)	(4, 0)
4P	(0, 1)	(0, 4)	(0, 1)	(0, 4)	O
5P	(0, 4)	(0, 1)	(2, 3)	(2, 2)	(0, 4)
6P	O	O	O	O	O

Как видно из приведенной таблицы, каждая точка характеризуется различным значением порядка: точки (0, 1) и (0, 4) имеют порядок 3, точки (2, 2) и (2, 3) – порядок 6, а точка (4, 0) – порядок 2.

Важным элементом, который характеризует эллиптическую кривую, является **генерирующая (базисная) точка *G***, порядок которой (n) должен являться, также как и число p, простым числом. Если умножать эту точку на числа меньшие, чем порядок точки, каждый раз будут получаться различные точки кривой.

Определение 5.5. Генерирующая (базисная) точка эллиптической кривой – это такая точка *G* на эллиптической кривой, для которой минимальное значение *n*, такое что $n \cdot G = O$, является *очень большим простым числом*.

Именно произведение точки G на большое случайное число используются как открытый параметр, а число – как тайный, например при реализации алгоритма Диффи – Хеллмана или ЭЦП (см. главу 12 настоящего пособия) на эллиптических кривых.

Выбор генерирующей точки обусловлен тем, чтобы ее порядок q был достаточно большим: обычно $2254 < q < 2256$. В табл. 5.11 приведены результаты выполнения операции $x \cdot G$ для кривой $E_{23}(10, 10)$ с генерирующей точкой $G = P(5, 1)$.

Таблица 5.11

Результаты выполнения операции $x \cdot G$ для кривой $E_{23}(10, 10)$ с генерирующей точкой $G = P(5, 1)$

x	1	2	3	4	5	6	7	8	9
$x \cdot G$	(5,1)	(8,21)	(11,5)	(10,11)	(12,8)	(7,20)	(15,19)	(9,1)	(9,22)
x	10	11	12	13	14	15	16	17	18
$x \cdot G$	(15,4)	(7,3)	(12,15)	(10,12)	(11,8)	(8,2)	(5,22)	O	(5,1)

Как видно, для выбранной генерирующей точки ее порядок равен 17, так как 17 является минимальным целым простым числом, для которого выполняется равенство $n \cdot P = O$.

5.4.4.2. Рекомендации по выбору параметров эллиптической кривой

Рассмотрим основные рекомендации по выбору параметров эллиптической кривой, предназначенной для решения криптографических задач, а именно задач по выбору коэффициентов a, b и модуля p. Основным критерием выбора является вероятность осуществления атак на криптосистему на основе эллиптической кривой.

Рассматриваемая ниже стратегия считается наиболее надежной с точки зрения обеспечения криптостойкости системы.

Изучим по шагам процесс формирования кривой.

Шаг 1. Выбираем случайно простое число p. Его длина в битах ($t = \log p + 1$; понятно, что t должно быть целочисленным с округлением в большую сторону) должна быть такой, чтобы сделать невозможным применение общих методов нахождения логарифмов на кривой. Величина t в настоящее время принимается не менее 256 битов.

Шаг 2. Выбираем случайные числа a и b такие, что

$$a, b \pmod p \neq 0 \text{ и } (4 \cdot a^3 + 27 \cdot b^2) \bmod p \neq 0.$$

Следует обратить внимание на то, что при вычислении композиции точек параметр b нигде не фигурирует. Поэтому для повышения эффективности счета рекомендуют случайно выбирать только b, а a принимать равным небольшому целому числу (например, –3).

Шаг 3. Определяем число точек N на кривой, равное $\#E_p(a, b)$. Важно, чтобы число N имело большой простой делитель q, а лучше всего чтобы само было простым числом: $N = q$. Если поиск кривой с $N = q$ занимает слишком много времени, то можно допустить $N = h \cdot q$, где h – небольшое число. Еще раз следует подчеркнуть, что стойкость криптосистемы на эллиптической кривой определяется не модулем p, а числом элементов q в подмножестве точек кривой. Но если множитель h – небольшое число, то q является величиной того же порядка, что и p. Если N не соответствует предъявляемым требованиям, то необходимо вернуться к шагу 2.

Шаг 4. Проверяем, выполняются ли неравенства $(p^k - 1) \bmod q \neq 0$ для всех k, $0 < k < 32$. Если нет, то возвращаемся к шагу 2. Эта проверка предотвращает возможность некоторых видов атак и исключает из рассмотрения так называемые суперсингулярные кривые и кривые с $N = p - 1$.

Шаг 5. Проверяем, выполняется ли неравенство $q \neq p$. Если нет, то возвращаемся к шагу 2. Для кривых с $q = p$, которые называются аномальными, существуют эффективные методы вычисления логарифмов, т. е. взлома шифра.

Шаг 6. Необходимая для криптографических приложений кривая уже получена. Имеем параметры p, a, b, количество точек N и размер подмножества точек q. Чтобы получить случайную точку на кривой, берем случайное число $x < p$, вычисляем $e \equiv (x^3 + a \cdot x + b) \bmod p$ и пытаемся извлечь квадратный корень $y = \sqrt{e} \bmod p$. Если корень существует, то получаем точку (x, y), в противном случае пробуем другое число x.

Задача, которую решает криптоаналитик при использовании криптосистемы на базе эллиптических уравнений, относится к задачам дискретного логарифмирования на эллиптической кривой и формулируется следующим образом: даны точки *P* и *Q* на эллиптической кривой порядка *N* (*N* – число точек на кривой). Необходимо найти единственное число *x*, такое что *P* = *x* · *Q*. Величина *x* и является дискретным логарифмом от *Q* по основанию *P*.

5.4.4.3. Система распределения криптографических ключей на основе эллиптической кривой

В качестве примера использования эллиптических кривых в криптографии рассмотрим систему распределения (обмена) тай-

ной ключевой информации. На основе кривых созданы и работают криптосистемы, использующие протоколы Эль-Гамаля, RSA и Шнорра[17].

Наша система работает подобно вышеописанному алгоритму Диффи – Хеллмана (см. пункт 5.6.1), т. е. предназначена для передачи закрытой информации по открытому каналу. Еще раз подчеркнем: стойкость рассматриваемого алгоритма определяется сложностью вычисления дискретного логарифма над эллиптической кривой $E_p(a, b)$. Действительно, вычисление точки Q как результата скалярного произведения целого положительного x на точку G является достаточно простой задачей. С другой стороны, вычисление значения x на основании точек Q и G является вычислительно трудной задачей. Для больших значений x эта задача практически неразрешима. На этом факте и основана рассматриваемая криптографическая система.

Обмен ключами с использованием эллиптических кривых может быть выполнен следующим образом.

Сначала выбирается простое число p и параметры a и b для эллиптической кривой. Это задает эллиптическую группу точек $E_p(a, b)$. Затем в $E_p(a, b)$ выбирается генерирующая точка $G = (x, y)$. При этом важно, чтобы наименьшее значение q, при котором выполняется условие $q \cdot G = O$, оказалось очень большим простым числом. Параметры $E_p(a, b)$ и G криптосистемы являются параметрами, известными всем участникам. Обмен ключами между участники информационного процесса (A и B) можно провести по следующему алгоритму.

1. Сторона A выбирает целое число k_a, меньшее q. Это число будет тайным ключом стороны A. Затем A генерирует открытый ключ: $Y_a = k_a \cdot G$, и отсылает его стороне B.

Открытый ключ представляет собой некоторую точку из $E_p(a, b)$.

2. Параллельно сторона B выбирает для себя тайный ключ k_b и вычисляет открытый ключ: $Y_b = k_b \cdot G$, и отсылает его стороне A.

3. Сторона A, получив открытый ключ стороны B, генерирует секретный ключ $K_a = k_a \cdot Y_b = K$, а сторона B соответственно генерирует секретный ключ $K_b = k_b \cdot Y_a = K$.

---

[17] Например, схема Шнорра используется в стандарте цифровой подписи в Республики Беларусь (см. главу 10).

Несложно установить, что обе стороны получили один и тот же результат, поскольку $k_a \cdot Y_b = k_a \cdot (k_b \cdot G) = k_b \cdot (k_a \cdot G) = k_b \cdot Y_a$.

Пример 5.13. Пусть $p = 211$; $G = (2, 2)$; $E_p(0, -4)$, что соответствует кривой $y^2 = x^3 - 4$.

Стороны совместно выбирают $q = 241$. Можно подсчитать, что $241 \cdot G = O$.

Тайным ключом стороны A является $k_a = 121$, поэтому открытым ключом стороны A будет $Y_a = 121(2, 2) = (115, 48)$.

Тайным ключом стороны B будет значение $k_b = 203$, поэтому открытым ключом стороны B будет $Y_b = 203(2, 2) = (130, 203)$.

Общим секретным ключом является одно значение, соответствующее точке на эллиптической кривой:

$$K = 121(130, 203) = 203(115, 48) = (161, 69).$$

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Запишите общий вид уравнения эллиптической кривой.
2. Продемонстрируйте графически порядок сложения двух точек эллиптической кривой.
3. Что такое «точка O»?
4. Запишите формально и представьте графически «закон удвоения точки» эллиптической кривой.
5. Чем определяется криптостойкость эллиптической криптографии?
6. Что такое «генерирующая точка» эллиптической кривой?
7. Сформулируйте задачу дискретного логарифмирования над эллиптическими кривыми.
8. Для ЭК $y^2 = (x^3 + a \cdot x + b) \bmod p$, параметры которой задаются таблицей, показать примеры операций сложения и умножения над точками (выбрать самостоятельно) кривой.

a	b	p
2	1	5
3	1	3
2	2	7
3	3	2
2	3	3
3	5	3
3	5	5

Глава 6
ПОТОКОВЫЕ ШИФРЫ

6.1. Общая характеристика потоковых шифров

Как мы убедились, в блочном шифре из двух одинаковых блоков открытого текста получаются одинаковые блоки шифрованного текста, что, безусловно, является одним из недостатков алгоритмов. Избежать этого позволяют поточные шифры, в которых шифрующее преобразование символа открытого текста меняется от одного элемента к другому.

Определение 6.1. ***Потоковый шифр*** **– это, по сути, симметричный шифр, в котором каждый символ открытого текста преобразуется в символ шифртекста *C* в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста *M*.**

Потоковый шифр реализует другой подход к симметричному шифрованию, нежели блочные шифры.

В качестве символов могут выступать как отдельные биты, так и символы (байты). Таким образом, потоковые шифры подходят для шифрования непрерывных потоков данных – голоса, видео и т. д.

Потоковые шифры преобразуют открытый текст в шифртекст *по одному биту* (*символу*) *за операцию*. *Генератор потока ключей* выдает поток битов: k_1, k_2, …, k_i. Этот поток битов (иногда называемый *бегущим ключом*) и поток битов открытого текст m_1, m_2, …, m_i, …, m_z подвергаются операции сложения по модулю два (XOR), и в результате получается поток битов шифртекста:

$$c_i = m_i \oplus k_i. \tag{6.1}$$

При расшифровании, для восстановления битов открытого текста, операция XOR выполняется над битами шифртекста и тем же самым потоком ключей:

$$m_i = c_i \oplus k_i. \tag{6.2}$$

Как видим, справедливо: $k_i = m_i \oplus c_i$.

Так как посимвольное шифрование при потоковом шифровании данных не приводит к задержкам в криптосистеме, то важ-

нейшее достоинство потоковых шифров – высокая скорость шифрования, соизмеримая со скоростью поступления входной информации.

Это позволяет обеспечить шифрование практически в реальном масштабе времени вне зависимости от объема информации и разрядности потока данных.

Классический пример потокового шифра – **шифр Вернама,** или **одноразовый блокнот**. Уже отмечалось, что если для гаммы последовательность битов выбирается случайно и длина гаммы равна, по крайней мере, длине сообщения, то взломать шифр невозможно. Но у данного режима шифрования есть и отрицательная особенность – проблемы с передачей и хранением ключей, ведь ключи, сравнимые по длине с передаваемыми сообщениями, трудно использовать на практике. Поэтому основная идея современных потоковых шифров – реализовать концепцию одноразового блокнота, используя секретный ключ меньшей длины, из которого для гаммы генерируется псевдослучайная числовая последовательность, похожая на случайную.

Безопасность системы полностью зависит от свойств генератора потока ключей. Если этот генератор выдает бесконечную строку нулей, шифртекст будет совпадать с открытым текстом и преобразование будет бессмысленным. Если генератор потока ключей выдает повторяющийся, например, 16-битный шаблон, криптостойкость системы будет пренебрежимо мала. В случае бесконечного потока случайных битов криптостойкость потокового шифра будет эквивалентна криптостойкости одноразового блокнота. Генератор потока ключей создает битовый поток, который похож на случайный, но в действительности обычно детерминирован и может быть с определенной вероятностью воспроизведен при расшифровании. Чем ближе выход генератора потока ключей к случайному, тем выше трудоемкость криптоаналитической атаки.

Блочные и потоковые шифры реализуются по-разному. Потоковые шифры, зашифровывающие и расшифровывающие данные по одному биту, не очень подходят для программных реализаций. С другой стороны, потоковые шифры больше подходят для аппаратной реализации.

Можно сказать, что зашифрование осуществляется *наложением гаммы* (шифрование *гаммированием*). А сама гамма является ключом шифрования. В практических поточных шифрах длина

шифртекста много больше длины секретного ключа, а ключевая последовательность является псевдослучайной и имеет некоторый период. Очевидно, что если последовательность битов гаммы не имеет периода и выбирается случайно, то взломать шифр невозможно. Если бы генератор выдавал бесконечную последовательность битов, в которой каждый бит порождался независимо и с вероятностью 1/2 принимал значения 0 или 1, то мы получили бы совершенно стойкий шифр. Но иметь ключ, равный по размеру шифруемым данным, представляется проблематичным. Поэтому потоковые шифры и вырабатывают выходную гамму на основе некоторого секретного ключа небольшого размера (например, 128 битов). С его помощью генерируется псевдослучайная гаммирующая последовательность (ПСГП). Она должна удовлетворять постулатам С. Голомба (всего три), основные из которых можно сформулировать следующим образом:

• количество «1» в каждом периоде ПСГП должно отличаться от количества «0» не более, чем на единицу;

• в каждом периоде ПСГП половина серий (из одинаковых символов) должна иметь длину один, одна четверть должна иметь длину два, одна восьмая должна иметь длину три и т. д; для каждой из этих длин должно быть одинаковое количество серий из «1» и «0».

Таким образом, **основной задачей потоковых шифров является выработка некоторой последовательности (выходной гаммы) для шифрования, т. е. выходная гамма является ключевым потоком для сообщения**.

В общем виде схема потокового шифра изображена на рис. 6.1.

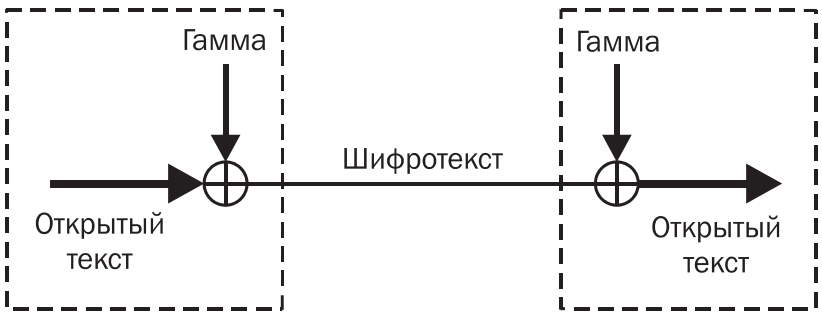


Рис. 6.1. Схема потокового шифра

Допустим, что в режиме гаммирования для потоковых шифров при передаче по каналу связи произошло искажение одного

символа шифртекста. Очевидно, что в этом случае все символы, принятые без искажения, будут расшифрованы правильно. Произойдет потеря лишь одного символа текста. А теперь представим, что один из символов шифртекста при передаче по каналу связи был потерян. Это приведет к неправильному расшифрованию всего текста, следующего за потерянным символом. Практически во всех каналах передачи данных для потоковых систем шифрования присутствуют помехи. Поэтому для предотвращения потери информации решают проблему синхронизации процедуры зашифрования и расшифрования текста. По способу решения этой проблемы шифрсистемы подразделяются на *синхронные* и *самосинхронизирующиеся* (*асинхронные*).

Проанализируем кратко особенности каждой системы.

6.2. Синхронные потоковые шифры

Определение 6.2. *Синхронные потоковые шифры (СПШ)* — шифры, в которых поток ключей генерируется независимо от открытого текста и шифртекста.

При зашифровании генератор потока ключей выдает биты потока ключей, которые идентичны битам потока ключей при расшифровании. Потеря символа шифртекста приведет к нарушению синхронизации между этими двумя генераторами и невозможности расшифрования оставшейся части сообщения. Очевидно, что в этой ситуации отправитель и получатель должны повторно синхронизироваться для продолжения работы.

Главное свойство СПШ – нераспространение ошибок. Ошибки отсутствуют, пока работают синхронно шифровальное и дешифровальное устройства отправителя и получателя информации. Перебои в работе системы называют *рассинхронизацией*. Ее может вызвать несовпадение скоростей зашифрования/расшифрования на разных концах канала связи, выпадение знаков при передаче и др. Если так произошло, то надо восстановить синхронность в работе генераторов гаммы – начать повторное шифрование с реинициализацией ключа обоими пользователями. Один из методов борьбы с рассинхронизацией – разбить отрытый текст на отрезки, начало и конец которых выделить вставкой *контрольных меток* (специ-

альных маркеров). В результате этого пропущенный при передаче символ приводит к неверному расшифрованию лишь до тех пор, пока не будет принят один из маркеров.

Синхронные потоковые шифры уязвимы к атакам на основе изменения отдельных бит шифртекста, **когда злоумышленник может изменить эти биты таким образом, что шифртекст расшифруется так, как это ему выгодно**.

Сформулируем в качестве итога краткого описания СПШ следующие их свойства:

• *требования по синхронизации*: при использовании СПШ получатель и отправитель должны быть синхронизированы, т. е. должны вырабатывать одинаковые значения ключевого потока для соответствующих символов передаваемого потока данных. Если синхронизация нарушится (например, вследствие потери символа при передаче), процесс расшифрования не даст корректного результата;

• *отсутствие размножения ошибок*: изменение символа шифртекста при передаче не вызывает ошибок при расшифровании других символов шифртекста;

• *свойство активной атаки*: любая вставка или удаление символа в шифртекст активным противником приводит к нарушению синхронизации и обнаруживается получателем, расшифровывающим сообщение (следствие первого свойства); активный противник может изменять символы шифртекста, и эти изменения приведут к соответствующим изменениям в открытом тексте, получаемом при расшифровании (следствие второго свойства).

6.3. Самосинхронизирующиеся потоковые шифры

Использование самосинхронизирующихся потоковых шифров (ССПШ) характерно для криптографии, используемой специальными ведомствами, и в литературе почти не освещается.

В самосинхронизирующихся потоковых шифрах символы ключевой гаммы зависят от исходного секретного ключа шифра и от конечного числа последних знаков зашифрованного текста.

Основная идея заключается в том, что внутреннее состояние генератора потока ключей является функцией предыдущих s битов

шифртекста. Поэтому генератор потока ключей на приемной стороне, приняв *s* битов, автоматически синхронизируется с шифрующим генератором.

Реализация ССПШ происходит следующим образом: каждое сообщение начинается случайным заголовком длиной *s* битов; заголовок шифруется, передается и расшифровывается; расшифровка является неправильной, зато после этих *s* битов оба генератора будут синхронизированы.

Недостаток этих потоковых шифров – распространение ошибок, так как искажение одного бита в процессе передачи шифртекста приведет к искажению нескольких битов гаммы.

Основные свойства ССПШ можно сформулировать следующим образом:

- *самосинхронизация*: должна наступать при удалении или вставке некоторых символов шифртекста, поскольку процесс расшифрования зависит от некоторого фиксированного числа предшествующих символов шифртекста. Это означает, что в случае удаления некоторого символа из шифртекста сначала будут выступать ошибки при расшифровании, однако затем установится автоматически требуемый режим и ошибок не будет;
- *ограниченное размножение ошибок:* состояние шифра зависит от *s* предыдущих символов шифртекста. Если во время передачи один символ шифртекста был изменен или удален/вставлен, то при расшифровке будет искажено не более *s* символов;
- *свойство активной атаки:* любое изменение символов шифртекста активным противником приведет к тому, что несколько символов шифртекста расшифруются неправильно, и это с большей (по сравнению с синхронными шифрами) вероятностью будет замечено со стороны получателя, расшифровывающего сообщение (следует из второго свойства). Однако в случае вставки или удаления знаков шифртекста (по первому свойству) это намного труднее обнаружить (по сравнению с синхронными шифрами, так как там наступает рассинхронизация). Поэтому необходимы дополнительные механизмы для контроля этой ситуации;
- *рассеивание статистики открытого текста:* поскольку каждый символ открытого текста влияет на весь последующий шифртекст, статистические свойства открытого текста не сохраняются в шифртексте.

6.4. Генераторы псевдослучайных последовательностей для потоковых шифров

6.4.1. Линейные конгруэнтные генераторы

Поскольку потоковый шифр максимально должен имитировать одноразовый блокнот, то шифрующая гамма (ключ) должна по своим свойствам максимально походить на случайную числовую последовательность. Такие последовательности вырабатываются специальными блоками систем потокового шифрования – *генераторами*. В действительности такие генераторы формируют *псевдослучайную последовательность* (ПСП, или *последовательность псевдослучайных чисел*), период которой имеет конечное значение. Проблема генерирования ПСП с максимально возможным периодом является основной при разработке и реализации систем потокового шифрования. Это, в частности, относится к генерированию бинарных ПСП. Часто период ПСП отождествляется с такой характеристикой генератора ПСП, как криптографическая стойкость.

Требования к криптографически стойкому генератору ПСП можно разделить на 2 группы: во-первых, они должны проходить статистические тесты на случайность (криптографически стойкий генератор ПСП должен удовлетворять «тесту на следуюший бит»), во-вторых, они должны сохранять непредсказуемость, даже если часть их исходного или текущего состояния становится известна криптоаналитику (генератор должен оставаться надежным даже в случае, когда часть или все его состояния стали известны (или были вычислены)).

Смысл указанного теста состоит в следующем*: не должно существовать полиномиального алгоритма, который на основе первых k битов случайной последовательности «сможет» предсказать (k + 1)-й бит с вероятностью более 0,5*.

Наиболее часто используемый алгоритм генерирования (программно или аппаратно) ПСП реализуется на основе так называемого *линейного конгруэнтного генератора*, описываемого следующим рекуррентным соотношением:

$$x_{t+1} = (a \cdot x_t + c) \bmod n, \qquad (6.3)$$

где x_t и x_{t+1} – соответственно t-й (предыдущий) и $(t + 1)$-й (текущий, вычисляемый) члены числовой последовательности; a, c и n – константы. Период такого генератора не превышает n.

При $c = 0$ получаем *мультипликативный конгруэнтный генератор* ПСП. *Примеры параметров* генератора на основе выражения (6.3) для регистров сдвига с 32-разрядной архитектурой: $n = 2^{31} - 1 = 2\,147\,483\,647$, $a = 16\,807$; 630 360 016; 10 783 183 814; 1 203 248 318; 397 204 094.

В сравнении с генераторами на основе выражения (6.3) более высокую криптостойкость обеспечивают подобные блоки, описываемые следующими соотношениями:

$$\left.\begin{aligned} x_{t+1} &= (1176 \cdot x_t + 1476 \cdot x_{t-1} + 1776 \cdot x_{t-2}) \bmod 2^{32} - 5, \\ x_{t+1} &= (2^{13} (x_t + x_{t-1} + x_{t-2}) \bmod 2^{32} - 5, \\ x_{t+1} &= (1995 \cdot x_t + 1998 \cdot x_{t-1} + 2001 \cdot x_{t-2}) \bmod 2^{32} - 849, \\ x_{t+1} &= (2^{19} (x_t + x_{t-1} + x_{t-2}) \bmod 2^{32} - 1629. \end{aligned}\right\} \quad (6.4)$$

Особенностями генератора, задаваемого формулами (6.3) и (6.4), является то, что начальное значение генератора x_0 является тайной; кроме того, если известно текущее (или предыдущие) значение ПСП, а также используемые константы, то криптоаналитику не представляет труда вычислить всю последовательность. В силу последнего обстоятельства данный тип генераторов не находит практического применения в серьезных ИС.

Комбинации нескольких (чаще двух) линейных конгруэнтных генераторов позволяют значительно повысить период ПСП. Б. Шнайер[18], например, приводит данные о том, как на 32-разрядных ПК реализовать составной генератор в виде комбинации двух генераторов, каждый из которых обеспечивает период соответственно $2^{31} - 85$ и $2^{31} - 249$, а комбинированный генератор позволяет достичь периода ПСП, равного произведению указанных чисел.

6.4.2. Генераторы ПСП на основе регистров сдвига с линейной обратной связью

Регистры сдвига (РС) используются в качестве генераторов ПСП в силу простоты реализации на основе цифровой логики. РС с линейной обратной связью (РСЛОС) состоит из двух частей: собственно РС и *функции обратной связи*. На рис. 6.2 представлена общая схема РС с линейной обратной связью. Функция обратной связи реа-

[18] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 816 с.

лизуется с помощью сумматоров сложения по модулю 2 (элементы XOR; на рис. 6.2 обозначены в виде кружочков со знаком сложения).

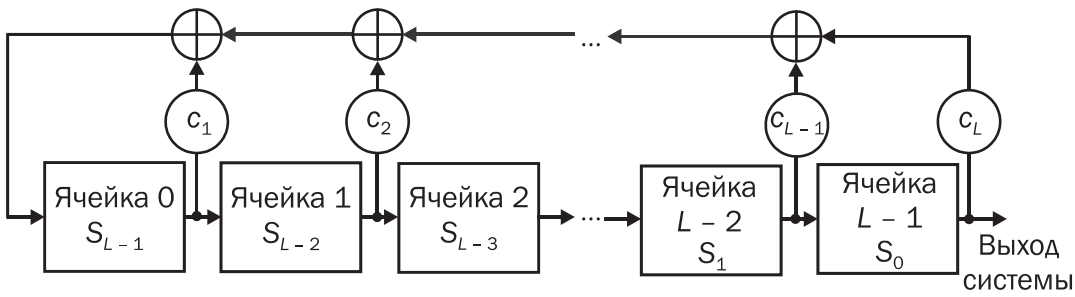


Рис. 6.2. Общая схема регистра сдвига с линейной обратной связью

Последний разряд РС в каждом такте формирует очередной символ псевдослучайной последовательности (ПСП). Такие последовательности называют также *М-последовательностями* (или последовательностями максимальной длины).

Количество ячеек L называют длиной регистра. Биты (ячейки) i обычно нумеруются числами от 0 до $L - 1$. Содержимое i-й ячейки обозначается через S_{L-1-i}.

В течение каждой единицы времени (за такт) выполняются следующие операции:

- содержимое ячейки $L - 1$ формирует часть выходной последовательности;
- содержимое i-й ячейки перемещается в $(i + 1)$-ю ячейку, новое содержимое ячейки 0 определяется битом обратной связи, который вычисляется сложением по модулю с определенными коэффициентами c_i битов ячеек.

Помимо этого, выходная последовательность зависит, главным образом, от начального состояния каждой ячейки регистра.

При аппаратной реализации РСЛОС отдельная ячейка может представлять собой триггер. В каждом новом такте состояние i-го триггера (его выходное состояние) передается на вход $(i + 1)$-го триггера и становится состоянием этого триггера.

РСЛОС строятся на основе *примитивных порождающих полиномов*, которые мы подробно анализировали при изучении циклических помехоустойчивых кодов. В частности, для схемы на рис. 6.2 такой полином, *или ассоциированный многочлен*, можно представить в следующем виде:

$$C(x) = 1 + c_1 \cdot x + c_2 \cdot x^2 + \ldots + c_L \cdot x^L. \qquad (6.5)$$

Его ненулевые коэффициенты c_i называются *отводами* (как и соответствующие ячейки регистра, составляющие значения аргументов функции обратной связи). В этом соотношении первый член суммы (1) означает, что результат вычислений в цепи обратной связи подается на вход 0-й ячейки.

Вспомним, что важным свойством многочлена $C(x)$ является приводимость. Многочлен называется *приводимым*, если он может быть представлен как произведение двух многочленов меньших степеней с коэффициентами из данного поля (в нашем случае с двоичными коэффициентами). Если нет, то многочлен называется *неприводимым*. Если многочлен является неприводимым, то период ПСП будет максимально возможным: $2^L - 1$. Иными словами: так как существует $2^L - 1$ разных ненулевых состояний регистра, то период последовательности, генерируемой РСЛОС при любом ненулевом начальном состоянии, не превышает $2^L - 1$.

Пример. Пусть задан многочлен, задающий структуру РСЛОС:

$$C(x) = 1 + c_1 \cdot x + c_3 \cdot x^3 = x^3 + x + 1;$$

как видим, для нашего случая коэффициенты полиномы равны следующим значениям: $c_3 = 1$, $c_2 = 0$, $c_1 = 1$.

В соответствии со значениями коэффициентов полиномы принято для краткости обозначать числами, соответствующими степеням ненулевых слагаемых полинома. Таким образом, для нашего случая это будет ряд 310. На рис. 6.3 представлена структурная схема, соответствующая заданному полиному.

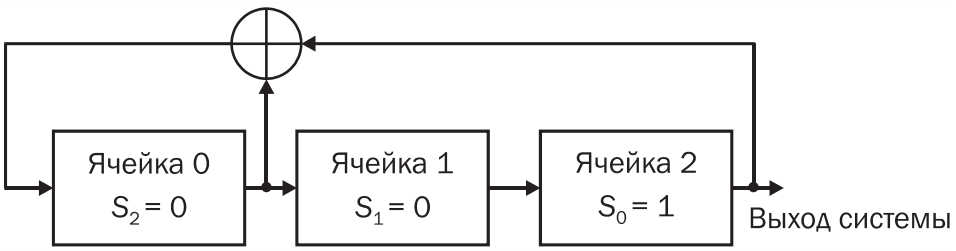


Рис. 6.3. Структурная схема РСЛОС, заданная полиномом $C(x) = 1 + c_1 \cdot x + c_3 \cdot x^3$

Положим, что начальное состояние ячеек регистра будет 001.

После первого такта работы генератора будут выполнены следующие операции:

1) содержимое ячейки 2 сформирует первый символ выходной последовательности (ПСП) генератора: этим символом будет 1;

2) в ячейку 0 будет записана сумма по модулю 2 начальных значений 0-й и 2-й ячеек: (0 + 1) mod 2 = 1;

3) в 1-ю ячейку будет записано состояние 0-й ячейки: 0;

4) во 2-ю ячейку будет записано состояние 1-й ячейки: 0.

Таким образом, после первого такта (цикла) работы генератора состояние его ячеек будет таким: 10**0** (жирным выделен очередной символ ПСП).

Легко вычислить состояние ячеек после второго цикла: 11**0**. После седьмого цикла состояние ячеек повторит начальное: 001. Это означает, что период ПСП равен 7. За эти 7 циклов будет сгенерирована следующая ПСП: 1001110.

Поскольку для рассматриваемого случая $L = 3$ и справедливо $2^L - 1 = 2^3 - 1 = 7$, то это является подтверждением тому, что полином

$$C(x) = 1 + c_1 \cdot x + c_3 \cdot x^3: (310)$$

является неприводимым.

Другими примерами неприводимых полиномов являются: 210, 320, 310, 410, 520, …, 84320, … .

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Поясните особенности потоковых шифров.
2. Дайте классификацию потоковых шифров.
3. Поясните достоинства и недостатки синхронных и самосинхронизирующихся потоковых шифров.
4. Нарисуйте общую структурную схему генератора ПСП на основе РЛОС.
5. Запишите в виде полинома следующую упрощенную форму его представления: 410, 520, 654321, 6420, 84320.
6. Представьте структурную схему регистра сдвига по заданию преподавателя.
7. Составьте таблицу состояний регистра сдвига из задания 6 и определите период генерируемой ПСП.

Раздел III

СТЕГАНОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

Глава 7
СТРУКТУРА, ОСОБЕННОСТИ ПОСТРОЕНИЯ И ИСПОЛЬЗОВАНИЯ СТЕГАНОГРАФИЧЕСКИХ СИСТЕМ

7.1. Терминология, сущность и цели стеганографического преобразования информации

Стеганография – это, в общем случае, искусство передачи скрытого сообщения. Принято считать, что впервые этот термин использовал Иоганн Тритемий (1462–1516) в своей работе под тем же названием: «Стеганография».

Слово «стеганография» в переводе с греческого буквально означает «тайнопись» (от греч. στεγανός (*steganos*) – скрытый + γράφω (*grafo*) – пишу). Местом зарождения стеганографии многие называют Египет, хотя первыми «стеганографическими сообщениями» можно назвать и наскальные рисунки древних людей. Первое упоминание о стеганографических методах в литературе приписывается Геродоту, который описал случай передачи сообщения Демартом. Он соскабливал воск с дощечек, писал письмо прямо на дереве, а потом заново покрывал дощечки воском.

Другой эпизод, который относят к тем же временам, – передача послания на голове раба. Для передачи тайного сообщения голову раба обривали, наносили на кожу татуировку и, когда волосы отрастали, отправляли с посланием.

В Китае письма писали на полосках щелка. Для сокрытия сообщений полоски с текстом письма сворачивались в шарики, покрывались воском и затем глотались посыльными.

Темное средневековье породило не только инквизицию: усиление слежки привело к развитию как криптографии, так и стега-

нографии. Именно в средние века впервые было применено совместное использование шифров и стеганографических методов.

XVII–XVIII века известны как эра «черных кабинетов» – специальных государственных органов по перехвату, перлюстрации и дешифрованию переписки. В штат «черных кабинетов», помимо криптографов и дешифровальщиков, входили и другие специалисты, в том числе и химики. Наличие специалистов-химиков было необходимо из-за активного использования так называемых невидимых чернил. Примером может служить любопытный исторический эпизод: восставшими дворянами в Бордо был арестован францисканский монах Берто, являвшийся агентом кардинала Мазарини. Восставшие разрешили Берто написать письмо знакомому священнику в город Блэй. Однако в конце этого письма религиозного содержания монах сделал приписку, на которую никто не обратил внимание: «Посылаю Вам глазную мазь: натрите ею глаза и Вы будете лучше видеть». Так он сумел переслать не только скрытое сообщение, но и указал способ его обнаружения. В результате монах Берто был спасен.

Еще древние римляне писали между строк невидимыми чернилами, в качестве которых использовались фруктовые соки, моча, молоко и некоторые другие натуральные вещества. Их опыт не был забыт. Стеганографические методы активно использовались в США в годы гражданской войны между южанами и северянами. Так, в 1779 году два агента северян Сэмюэль Вудхулл и Роберт Тоунсенд передавали информацию Джорджу Вашингтону, используя специальные чернила.

Различные симпатические чернила использовали и русские революционеры в начале XX века. Симпатические чернила или с успехом выполняющее их роль обычное молоко – один из самых распространенных стеганографических методов. Многие книги Ленина были написаны в местах заключения молоком между строк. Чернильницей Владимиру Ильичу служил хлебный мякиш – при малейшем подозрительном звуке В. И. Ленин съедал свои приспособления. Позднее исписанные молоком листы передавались на волю, а там нагревались над лампой и переписывались товарищами по партии. Впрочем, царская охрана тоже знала об этом методе (в ее архиве хранится документ, в котором описан способ использования симпатических чернил и приведен текст перехваченного тайного сообщения революционеров).

К середине XX века стеганография достигла значительных успехов, чему не мало поспособствовали Первая и Вторая мировые войны. Особенных успехов добились немцы, которые во время Второй мировой войны широко применяли «микроточки», представлявшие из себя микрофотографии размером с обычную типографскую точку. При увеличении «микроточка» давала четкое изображение печатной страницы стандартного размера. Такая точка или несколько точек вклеивались в обыкновенное письмо и, помимо сложности обнаружения, обладали способностью передавать большие объемы информации, включая чертежи и рисунки.

Сам метод был придуман намного раньше. Микроточки появились сразу же после изобретения Луи Дагером фотографического процесса и впервые в военном деле были использованы во времена франко-прусской войны (в 1870 году), но широкого применения до Второй мировой войны этот метод не имел. Но во время Второй мировой войны данный метод претерпел второе рождение и успех его был весьма заметным. Американцы, впечатленные достижениями своего противника в стеганографии, после войны запретили даже такие относительно невинные операции, как пересылку посредством почты записей шахматных партий, инструкций по вязанию и даже детских рисунков как наиболее простых с точки зрения стеганографа объектов для встраивания шпионских сообщений[19].

Таким образом, *стеганография* – это наука о способах передачи (хранения) сокрытой информации, при которых скрытый канал организуется на базе и внутри открытого канала с применением особенностей восприятия информации, причем для этой цели могут использоваться следующие приемы:

– полное сокрытие факта существования скрытого канала связи;

– создание трудностей для обнаружения, извлечения или модификации передаваемых сокрытых сообщений внутри открытых сообщений-контейнеров;

– маскировка сокрытой информации в протоколе.

Компьютерная стеганография изучает способы сокрытия информации в компьютерных данных, представляющих собой различные файлы, программы, пакеты протоколов и т. п.

---

[19] Компьютерная стеганография // Все о хакерах, хакинге и защите информации [Электронный ресурс]. URL: http://supermegayo.ru/compterr/77.html (дата обращения: 16.12.2011).

Современная стеганография, как правило, имеет дело с электронными средствами. Компьютерная стеганография базируется на двух принципах. Первый заключается в том, что файлы, содержащие оцифрованное изображение или звук, могут быть до некоторой степени видоизменены без потери функциональности, в отличие от других типов данных, требующих абсолютной точности.

Второй принцип состоит в неспособности органов чувств человека различить незначительные изменения в цвете изображения или качестве звука, что особенно легко использовать применительно к объекту, несущему избыточную информацию, будь то 16-битный звук, 8-битное или, еще лучше, 24-битное изображение. Если речь идет об изображении, то изменение значений наименее важных битов, отвечающих за цвет пикселя, не приводит к сколь-нибудь заметному для человека изменению цвета.

Особенностью стеганографического подхода является то, что он не предусматривает прямого оглашения факта существования защищаемой информации. Это обстоятельство позволяет в рамках традиционно существующих информационных потоков или информационной среды решать некоторые важные задачи защиты информации ряда прикладных областей.

Основным определяющим моментом в стеганографии является *стеганографическое преобразование*. До недавнего времени стеганография, как наука, в основном изучала отдельные методы сокрытия информации и способы их технической реализации. Разнообразие принципов, заложенных в стеганографических методах, по существу, тормозило развитие стеганографии как отдельной научной дисциплины и не позволило ей сформироваться в виде некоторой науки со своими теоретическими положениями и единой концептуальной системой, которая обеспечила бы формальное получение качественных и количественных оценок стеганометодов. В этом история развития стеганографии резко отличается от развития криптографии.

Как и любое новое направление, компьютерная стеганография, несмотря на большое количество открытых публикаций и ежегодные конференции, долго не имела единой терминологии. На конференции Information Hiding: First Information Workshop в 1996 году было предложено использовать единую терминологию и оговорены основные термины[20].

[20] Pfitzmann B. Information Hiding Terminology // Springer Lecture Notes in Computer Science. 1996. Vol. 1174. P. 347–350.

Определение 7.1. ***Стеганографическая система*** **(стегосистема[21]) – совокупность средств и методов, которые используются для формирования скрытого канала передачи информации.**

Стегосистема образует стегоканал, по которому передается (или в котором хранится) заполненный контейнер. Этот канал считается подверженным воздействиям со стороны нарушителей.

При построении стегосистемы должны учитываться следующие положения:

– противник имеет полное представление о стеганографической системе и деталях ее реализации; единственной информацией, которая остается неизвестной потенциальному противнику, является ключ, с помощью которого только его держатель может установить факт присутствия и содержание скрытого сообщения;

– если противник каким-то образом узнает о факте существования скрытого сообщения, это не должно позволить ему извлечь подобные сообщения в других данных до тех пор, пока ключ хранится в тайне;

– потенциальный противник должен быть лишен каких-либо технических и иных преимуществ в распознавании или раскрытии содержания тайных сообщений.

Любая стегосистема должна отвечать следующим требованиям:

– *свойства контейнера должны быть модифицированы*, чтобы изменение невозможно было выявить при визуальном контроле. Это требование определяет качество сокрытия внедряемого сообщения: для обеспечения беспрепятственного прохождения стегосообщения по каналу связи оно никоим образом не должно привлечь внимание атакующего;

– *стегосообщение должно быть устойчиво к искажениям*, в том числе и злонамеренным. В процессе передачи изображение (звук или другой контейнер) может претерпевать различные трансформации: уменьшаться или увеличиваться, преобразовываться в другой формат и т. д. Кроме того, оно может быть сжато, в том числе и с использованием алгоритмов сжатия с потерей данных;

---

[21] В литературе встречаются два сокращенных термина, обозначающие саму стеганографическую систему или ее элементы: ***стего****система* и ***стега-но****система*.

– для *сохранения целостности встраиваемого сообщения* необходимо использование кода с исправлением ошибки;

– для *повышения надежности встраиваемое сообщение должно быть продублировано*.

Кратко охарактеризуем другие основные понятия, относящиеся к предметной области: *сообщение*, *контейнер* и *ключ*.

Термин «контейнер» употребляется большинством авторов, поскольку является дословным переводом устоявшегося английского термина «container», обозначающего несекретную информацию, которую используют для сокрытия сообщений. По сути же контейнер в стеганографической системе является не чем иным, как носителем сокрытой информации, поэтому вполне возможно использование и такого термина. В некоторых источниках термин «контейнер» заменяют названием «стего», который также является производным от английского сокращения «stego» (полное название «stegano»).

Определение 7.2. *Контейнером* (носителем) *C* называют несекретные данные, которые используют для сокрытия сообщений (рис. 7.1).

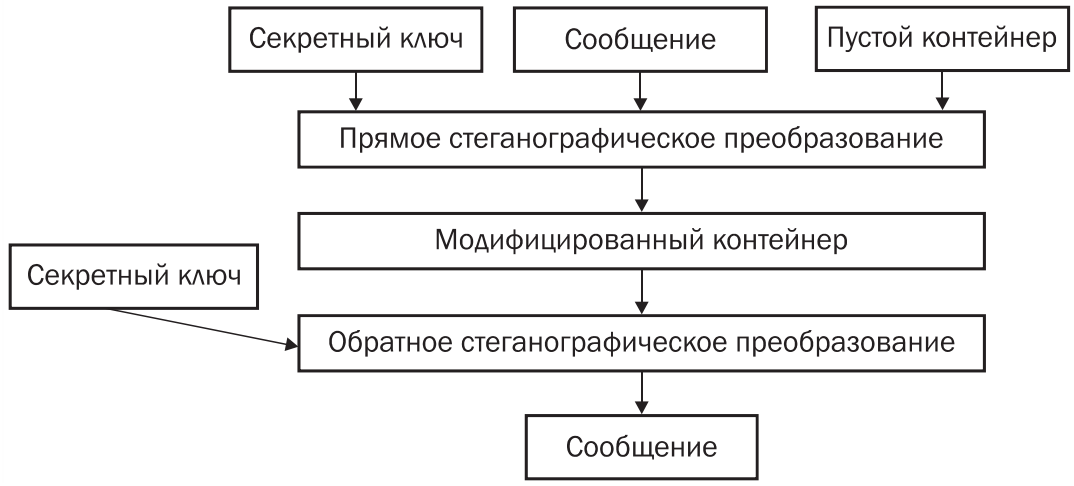


Рис. 7.1. Общая схема стеганографического преобразования информации

Пустой контейнер (немодифицированный контейнер) – это некоторый контейнер *C*, не содержащий сообщения. *Заполненный контейнер* (модифицированный контейнер) – это элемент *C* системы, содержащий сообщение *M*.

В компьютерной стеганографии в качестве контейнеров могут быть использованы различные оцифрованные данные: растровые графические изображения, цифровой звук, цифровое видео, всевозможные носители цифровой информации, а также текстовые и другие электронные документы.

Выделяют два типа контейнера: *потоковый* и *фиксированный*.

Потоковый контейнер представляет собой последовательность битов, которая непрерывно изменяется. Особенностью такого контейнера является то, что невозможно определить его начало или конец. В непрерывном потоке данных самая большая трудность для получателя – определить, когда начинается скрытое сообщение. Сообщение встраивается в контейнер в реальном масштабе времени, поэтому в кодере заранее неизвестно, хватит ли размеров контейнера для передачи всего сообщения. В то же время в один контейнер большого объема может быть встроено несколько сообщений.

В фиксированном контейнере известны его размеры и характеристики. Это позволяет выполнять встраивание данных оптимальным способом.

Контейнер может быть *избранным*, *случайным* или *навязанным*. Избранный контейнер зависит от встроенного сообщения, а в предельном случае является его функцией. Такой тип контейнера характерен именно для стеганографии. Навязанный контейнер появляется, когда тот, кто предоставляет контейнер, подозревает о возможной скрытой переписке и желает предотвратить ее. На практике чаще всего имеют дело со случайным контейнером.

Скрытие информации большого объема выдвигает существенные требования к контейнеру, размер которого должен, по меньшей мере, в несколько раз превышать размер встраиваемых данных. Для увеличения скрытости указанное соотношение должно быть как можно большим.

Перед тем как выполнить встраивание сообщения в контейнер, его необходимо преобразовать в определенный удобный для упаковки вид. Кроме того, для повышения защищенности секретной информации последнюю можно зашифровать достаточно устойчивым криптографическим ключом.

Определение 7.3. ***Сообщением (стегосообщением) M*** **называют секретную информацию, наличие которой в контейнере необходимо сокрыть.**

Определение 7.4. ***Ключом (стегоключом) K*** **называют секретную информацию, известную только законному пользователю, которая определяет конкретный вид алгоритма сокрытия.**

Через K обозначается множество всех допустимых секретных ключей. Заметим, что в работах по стеганографии ключ понимается как в широком, так и в узком смысле. В широком смысле стеганографический ключ – это сам неизвестный противнику способ сокрытия информации.

При этом необходимо отличать стеганографический ключ от криптографического, который также может присутствовать в системе и использоваться для предварительного криптографического закрытия внедряемой информации.

По аналогии с криптографией, по типу стегоключа стеганографические системы можно подразделить на три типа: *с секретным ключом*; *с открытым ключом*; *смешанные*.

В стегосистеме с секретным ключом используется один ключ, который должен быть либо определен до начала обмена секретными сообщениями, либо передан по защищенному каналу.

По известному нам постулату Керкгоффса, безопасность системы должна базироваться на определенном фрагменте секретной информации – ключе, который (как правило, предварительно) разделяется между авторизованными лицами. Отправитель, встраивая секретное сообщение в избранный контейнер C, использует стегоключ K. Если получатель знает данный ключ, то он может извлечь из контейнера секретное сообщение. Без знания ключа любое постороннее лицо этого сделать не сможет.

Необходимо отметить, что в некоторых алгоритмах во время извлечения скрытой информации дополнительно необходимы сведения о первичном (пустом) контейнере или некоторые другие данные, отсутствующие в стеганограмме. Такие системы представляют ограниченный интерес, поскольку они требуют передачи изначального вида контейнера, что эквивалентно традиционной задаче обмена ключами.

Для функционирования стеганографической системы с открытым ключом необходимо иметь два стегоключа: один секретный, который нужно хранить в тайне, а другой открытый, который может храниться в доступном для всех месте и может пере-

даваться свободно по незащищенному каналу связи. Ключи различаются таким образом, что с помощью вычислений невозможно вывести один ключ из другого. При этом открытый ключ используется для встраивания сообщения, а секретный – для его извлечения. Следует отметить, что стегоключ не шифрует данные, а скрывает место их встраивания в контейнере. Кроме того, данная схема хорошо работает и при взаимном недоверии отправителя и получателя.

Для функционирования бесключевых стегосистем отсутствует необходимость в дополнительных данных (наподобие стегоключа).

На практике преимущество отдается именно бесключевым стегосистемам, хотя они могут быть раскрыты в случае, если нарушитель узнает о методе стеганопреобразования, который был при этом использован. В связи с этим в бесключевых системах часто используют особенности криптографических систем с открытым и/или секретным ключом.

7.2. Основные направления использования стеганографических систем

Современная стеганография, как правило, «имеет дело» с электронными средствами. Это может объясняться следующими причинами. Во-первых, так как объем осаждаемой информации, как правило, довольно небольшой по сравнению с размером контейнера, в котором она будет скрыта, то в электронных контейнерах гораздо проще скрывать данные и извлекать их. Во-вторых, процедура осаждения/извлечения может быть автоматизирована с помощью специальных программных средств. В-третьих, электронный формат данных характеризуется информационной избыточностью, которой можно управлять, чтобы скрыть сообщения. Эти общие для всех электронных документов особенности, на наш взгляд, ставят знак равенства между цифровой и компьютерной стеганографией.

На рис. 7.2 приведена наиболее общая классификация методов. Мы приводим ее здесь (в дополнение к вышеуказанной классификации на основе типа контейнера) для лучшего понимания предметной области.

Рис. 7.2. Общая классификация методов компьютерной стеганографии

Использование стеганографических систем является наиболее эффективным при решении *проблемы защиты информации с ограниченным доступом*. Это означает возможность скрытой передачи информации.

Кроме указанного направления, **стеганография является одним из перспективных средств *для аутентификации и маркировки авторской продукции* с целью защиты авторских прав на цифровые объекты от пиратского копирования**.

На компьютерные графические изображения, аудиопродукцию, литературные произведения (программы в том числе) наносится специальная метка, которая остается невидимой для глаз, но распознается специальным программным обеспечением.

Метка содержит скрытую информацию, подтверждающую авторство. Скрытая информация призвана обеспечить защиту прав интеллектуальной собственности.

В качестве внедряемой информации можно использовать данные об авторе, дату и место создания произведения, номера документов, подтверждающих авторство, дату приоритета и т. д. В качестве иллюстрации более подробно рассмотрим использование стеганографии для защиты авторских прав на текстовые документы.

7.2.1. Методы текстовой стеганографии

Компьютерная стеганография базируется на двух принципах. Первый заключается в том, что если файлы, содержащие оцифрованное изображение или звук, могут быть до некоторой степени видоизменены без потери функциональности, то текстовые документы, коды программ или базы данных требуют абсолютной точности при обратных преобразованиях. Это обстоятельство чрезвычайно важно, если, например, текстовый документ-контейнер с осажденной информацией претерпевает конвертацию на основе иного стиля (шрифта, кегля и т. п.) или при его архивации. Понятно, что сам документ при этом не должен измениться.

Все многообразие методов текстовой стеганографии подразделяется на *синтаксические*, которые не затрагивают семантику текстового сообщения, и *лингвистические* методы, которые основаны на эквивалентной трансформации текстовых файлов, сохраняющей смысловое содержание текста, его семантику.

7.2.1.1. Синтаксические методы текстовой стеганографии

Line-shift coding (изменение расстояния между строками электронного текста). Он также называется *методом изменения межстрочных интервалов*. Его сущность заключается в том, что используется текст с различными межстрочными расстояниями. Выделяется максимальное и минимальное расстояния между строками, позволяющее кодировать соответственно символы «1» и «0» осаждаемого сообщения (рис. 7.3). Разница в межстрочных расстояниях авторами изменялась на 1/300 дюйма (это расстояние было привязано к существовавшей в то время разрешающей способности монитора). Очевидным недостатком метода является его низкая эффективность: размер в битах осаждаемой информации не может превысить количество строк в контейнере.

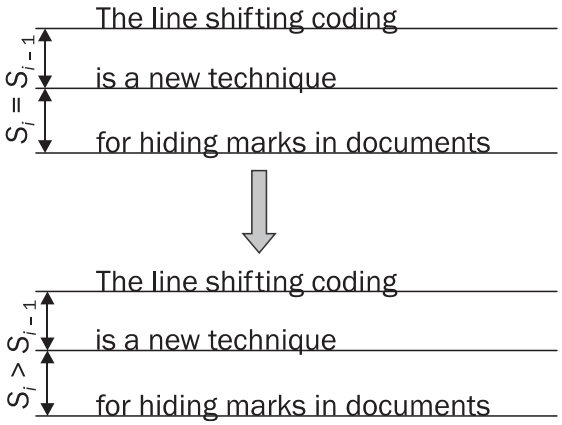


Рис. 7.3. Иллюстрация метода
на основе изменения межстрочных интервалов

Word-shift coding (изменение расстояния между словами в одной строке электронного текста). Суть метода состоит в том, что осаждение информации основано на модификации расстояния между словами текста-контейнера. Аналогично предыдущему методу, выделяется максимальное и минимальное расстояния между словами, обозначающие соответственно символ «1» и «0», и остальные расстояния, или некоторые из них, увеличивают или уменьшают до размеров уже выделенных. Частный случай этого метода – *метод изменения количества пробелов* (рис. 7.4). Данный рисунок показывает пример внедрения в текст-контейнер бинарной последовательности 0101100100111010. Как видно, переход с одинарного пробела на двойной кодирует «1», переход же с двойного пробела на одинарный кодирует «0».

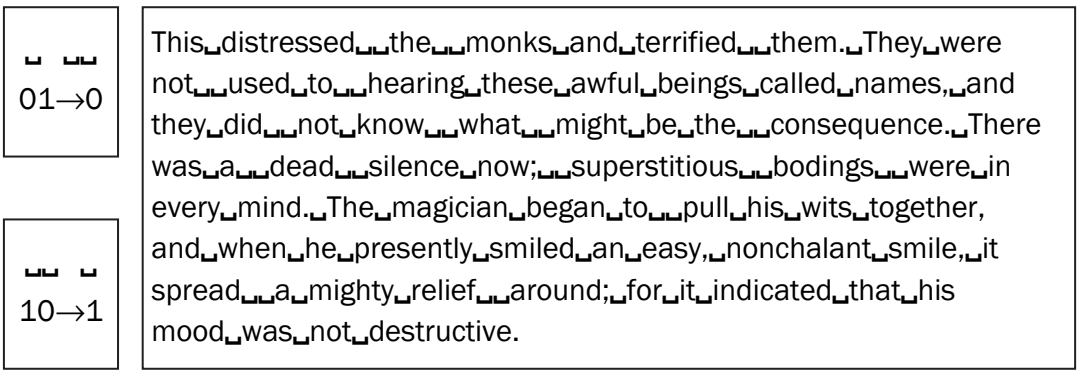


Рис. 7.4. Пример использования метода
на основе длин пробелов

Данный метод имеет недостатки. Во-первых, он мало эффективен, так как необходим контейнер большого объема (объем скрытых данных в данном случае приблизительно равен одному биту на 160 байт текста). Во-вторых, возможность сокрытия зависит от структуры текста (некоторые тексты, например белые стихи, не имеют четких признаков конца). В-третьих, текстовые редакторы часто автоматически добавляют символы пробела после точки.

Feature coding (внесение специфических изменений в шрифты (начертания отдельных букв)). Этот метод заключается в изменении написания отдельных букв используемого стандартного шрифта (рис. 7.5).

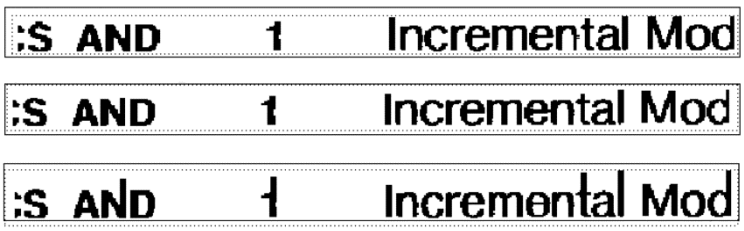


Рис. 7.5. Иллюстрация метода *feature coding*

Визуально заметны различные образы, соответствующие буквам с верхними (например, *l*, *t*, *d*) или нижними (например, *a*, *g*) выносными элементами. Так, букву «А» можно модифицировать, незначительно укорачивая длинную нижнюю часть буквы. При этом можно закодировать стегосообщение так, что модифицированная буква будет означать «1», а немодифицированная – «0».

Модифицировать можно несколько букв. Таким образом, объем встраиваемого сообщения будет увеличиваться.

Результат внедрения секретного сообщения «1» в текст-контейнер «А», при использовании метода *feature coding* и текстового процессора MS Office Word 2007, показан на рис. 7.6.

а *б*

Рис. 7.6. Пример применения метода *feature coding*:
а – пустой контейнер; *б* – заполненный контейнер (со стегосообщением «1»)

Известны иные методы текстовой стеганографии. Кратко охарактеризуем их.

Метод изменения интервала табуляции. Аналогичен вышеописанному методу изменения количества пробелов, только в этом случае меняется не количество пробелов, а соответственно расстояние между строками и интервал табуляции.

Null chipper (дословно – несуществующий, нулевой лепет). Предполагает размещение тайной информации на установленных позициях слов или в определенных словах текста-контейнера, который, как правило, лишен логического смысла (как видно, действительно лепет). На рис. 7.7 показан пример реализации метода – скрытой информацией являются первые символы слов.

«President's embargo ruling should have immediate notice. grave situation affecting international law. statement foreshadows ruin of many neutrals. yellow journals unifying national excitement immensely».

Pershingsailsfromnyjunei → Pershing Sails From NY June I.

Рис. 7.7. Пример реализации метода *null chipper*

Метод увеличения длины строки. Предусматривает искусственное увеличение длины каждой строки за счет пробелов: например, одному пробелу соответствует логический 0, двум – 1 (рис. 7.8).

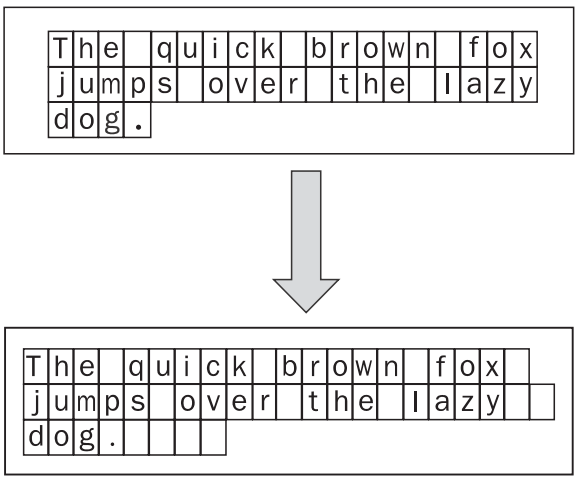


Рис. 7.8. Иллюстрация метода увеличения длины строки

Преимущество такого метода кодирования состоит в том, что оно может быть выполнено с любым текстом; изменения в фор-

мате резко не бросаются в глаза читателю, обеспечивается передача большего числа скрытых данных по сравнению с предыдущим методом (примерно 1 бит на 80 байт содержимого контейнера). Недостаток метода состоит в том, что некоторые компьютерные программы (например, *Sendmail*) могут неосторожно удалять дополнительные пробелы. Помимо этого, скрытые таким образом данные не всегда могут быть восстановлены с печатной копии документа.

Использование регистра букв. Для обозначения бита секретного сообщения, представленного единицей, используется *символ* нижнего регистра, а нулем – верхнего (или наоборот). Например, секретный текст, состоящий из одной буквы «A», необходимо внедрить в текст-контейнер «steganography». Для этого используем двоичное представление кода символа «A» – «01000001». Далее предположим, что для обозначения бита секретного сообщения, представленного единицей, используется символ верхнего регистра, а нулем – нижнего.

Результат внедрения секретного сообщения «A» в текст-контейнер «steganography» показан на рис. 7.9.

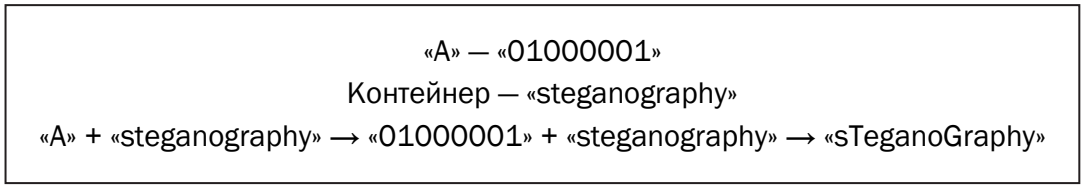


Рис. 7.9. Иллюстрация метода на основе регистра клавиатуры

Существует модификация данного метода, основанная на применении различных алфавитов, в которых используются символы, имеющие одинаковое начертание, но различную кодировку (например, *a*, *e*, *p*, *m*, *c*).

Метод невидимых символов. Пробел кодируется символом с кодом 32, но в тексте его можно заменить также символом, имеющим код 255 (или 0), который является «невидимым» и отображается как пробел.

Методы могут применяться независимо и совместно, сохраняют исходный смысл текста, а обеспечиваемые ими показатели плотности кодирования при совмещении складываются.

Рассмотренные методы работают успешно до тех пор, пока тексты представлены в коде ASCII. Существуют также стегано-

графические методы, которые интерпретируют текст как двоичное изображение. Необходимо отметить, что *данные методы нечувствительны к изменению масштаба документа, что обеспечивает им хорошую устойчивость к большинству искажений, которые могут иметь место при активных атаках*.

Описанные выше методы – синтаксические – легко применяются к любому тексту, независимо от его содержания, назначения и языка. *Синтаксические системы стеганографии легко реализуются в программном коде*, так как они полностью автоматические и не требуют вмешательства оператора. Однако синтаксические методы неустойчивы к форматированию текста (вспомним робастность систем на основе цифрового водяного знака (ЦВЗ)), и поэтому информация может быть потеряна при простом применении иного стиля форматирования текста-контейнера, скрывающего в себе стегосообщение. К тому же с помощью синтаксических методов можно передать незначительное количество информации. В литературе отсутствуют результаты экспериментального исследования эффективности проанализированных методов с проверкой на больших объемах данных.

7.2.1.2. Лингвистические методы текстовой стеганографии

Стеганографические методы, основанные на лексической структуре текста, обладают большими возможностями.

Лингвистическая стеганография занимается скрытым кодированием произвольной информации, представленной в двоичном виде, в текстах.

Необходимо отметить, что осмысленность и внешняя «безобидность» текста должна сохраниться. К *лексическим* методам встраивания скрытой информации в текстовые файлы-контейнеры относят *метод переменной длины слова, метод первой буквы, метод синонимов* и другие.

Одним из наиболее обсуждаемых методов является **метод**, **основанный на системе синонимов языка**, используемого для написания электронного текста. Проведенные исследования для случая английского языка показали, что среднее количество синонимов в одном подмножестве синонимов равняется 2,56. Минимальное количество синонимов в одном множестве синонимов равняется 2, а максимальное 13.

Пример 7.1. В качестве примера приведем множество синонимов S_0: {«propensity», «predilection», «penchant», «proximity»}. В приведенном множестве синонимов каждое слово имеет единственное одинаковое смысловое значение, что позволяет закодировать каждое слово своим уникальным кодом, например «propensity» – 00, «predilection» – 01, «penchant» – 10, «proximity» – 11. Подобное кодирование позволяет выбирать одно из четырех слов в зависимости от двух бит секретного сообщения. Отметим, что при этом, независимо какое из четырех слов будет выбрано, семантика (смысл) сообщения не изменится.

Процедура передачи секретного сообщения с использованием лексической стеганографии производится в следующей последовательности.

Отправитель и получатель имеют одинаковое множество синонимов, поддерживаемое одним и тем же электронным словарем.

1. Первоначально отправитель выбирает контейнер (текстовый файл).

2. Отправитель преобразует секретное сообщение в двоичную последовательность: ...01000... , используя по возможности криптографические методы.

3. Отправитель, последовательно анализируя текстовый файл, находит первое слово, для которого существуют N синонимов.

4. Отправитель вычисляет целую часть значения $\log_2 N$, определяющую число символов секретного сообщения, которые могут быть внедрены в контейнер путем выбора соответствующего синонима. Например, если в тексте встретилось слово «penchant», принадлежащее множеству S_0: {«propensity», «predilection», «penchant», «proximity»}, в соответствии со значением двух бит закодированной секретной информации – 01, это слово должно быть заменено словом синонимом «predilection».

Аналогичные действия выполняет и получатель. Получатель анализирует слова в контейнере (текстовом файле) на предмет принадлежности к множеству синонимов. Если текущее слово относится к одному из множеств синонимов, он определяет мощность этого множества N. Целая часть $\log_2 N$ определяет число битов, которые закодированы на основании текущего множества синонимов. Например, если получатель обнаружит в тексте слово «predilection» и определит, что оно относится к множеству синонимов S_0, состоящему из $N = 4$ синонимов, тогда $\log_2 N = 2$ бита, а

слово «predilection» интерпретируется как два бита (01) секретного сообщения. Следует отметить, что в каждом подмножестве синонимов их упорядочивание должно выполняться по одному и тому же алгоритму и у отправителя сообщения, и у его получателя (например, в алфавитном порядке).

В случае слов с несколькими смысловыми значениями подобное кодирование оказывается невозможным. Также невозможно кодирование, если один из синонимов состоит из двух (или более) разделенных пробелом слов.

К сожалению, количество синонимов не всегда равно 2^k, где k – целое положительное число. Например, имеем три подмножества синонимов S_0: {AAA, BBB, CCC}; S_1{MMM, NNN, OOO, PPP, QQQ}; S_2:{WWW, XXX, YYY}.

Использование данных подмножеств синонимов, согласно приведенной ранее идее семантической стеганографии, позволяют закодировать один бит на основании первого и третьего подмножеств и два бита за счет второго подмножества. Всего может быть закодировано только четыре бита, в то время как суммарная мощность приведенных подмножеств позволяет закодировать $3 \cdot 5 \cdot 3 = 45$ состояний или не менее, чем $|\log_2 45| = 5$ бит.

Для увеличения объема внедряемой в контейнер секретной информации может быть использована система счисления со смешанным основанием. В этом случае работают не с одним множеством синонимов, а с группой множеств синонимов. Кодирование выполняется не путем изолированного использования каждого множества синонимов, а используя их совокупности. Например, для приведенных **трех множеств** можно закодировать $\log_2 45 = 5$ битов (берем, понятно, целочисленное значение) вместо 4. Для этого используем код, состоящий из трех цифр, в котором первая и последняя цифры могут принимать значения от 0 до 2, а средняя – от 0 до 4.

Отличительной особенностью методов *лексической стеганографии* является то, что пользователь, как правило, должен сам составлять (или видоизменять) тот объект (текст-контейнер), в котором будет передаваться или храниться тайная информация.

Так, при использовании **метода переменной длины** пользователю, который хочет послать секретное сообщение, необходимо сгенерировать (набрать) текст, в котором слова должны иметь соответствующую длину. Длина этих слов зависит от секретного со-

общения и способа кодирования. Обычно одно слово текста-контейнера определенной длины кодирует два бита информации из стегосообщения. Например, слова текста длиной в 4 и 8 символов могут означать комбинацию бит «00», длиной в 5 и 9 – «01», 6 и 10 – «10», 7 и 11 букв – «11». Слова короче 4 и длиннее 11 букв можно вставлять где угодно для лексической и грамматической связки слов в предложении. Программное приложение, которое декодирует принятое сообщение, будет просто игнорировать их.

При использовании **метода первой буквы** можно передавать еще больше скрытой информации в одном слове: обычно это три или четыре бита. Программа-помощник в этом методе накладывает ограничение уже не на длину слова, а на первую (можно на вторую) букву. Обычно одну и ту же комбинацию могут кодировать несколько букв, например, комбинацию «101» означают слова, начинающиеся с «А», «Г» или «Т». Это дает большую свободу выбора оператору, придумывающему стегосообщение, и текст не будет нелепым и не содержащим смысла.

Другим, не менее распространенным лексическим методом передачи скрытой информации является **мимикрия**. Мимикрия генерирует осмысленный текст, используя синтаксис, описанный в **Context Free Grammar** (CFG), и встраивает информацию, выбирая из CFG определенные фразы и слова. Грамматика CFG – это один из способов описания языка, который состоит из статических слов и фраз языка, а также узлов. Узлы в простейшем случае представляют собой места в генерируемом тексте, где может быть принято решение, какое слово или фразу дальше необходимо вставить в текст. Мимикрия создает бинарное дерево, которое основано на возможностях CFG, и составляет текст, выбирая те листья дерева, которые кодируют нужный бит.

Пример 7.2. Необходимо передать секретное сообщение 10101, используя следующее бинарное дерево:

Старт → существительное
существительное → Илья || Иван
глагол → поехал куда || пошел куда
куда → на работу, чтобы зачем || домой, чтобы зачем
зачем → забрать что || взять что
что → деньги || одежду.

Основываясь на приведенном бинарном дереве, покажем процедуру внедрения секретного кода «10101» (табл. 7.1).

Таблица 7.1

Внедрение секретного кода на основе бинарного дерева

Узлы принятия решения	Скрытый бит	Ответы, полученные в результате внедрения кода 10101
Старт	–	старт → существительное
Существительное	1	существительное → Иван
Иван глагол	0	глагол → поехал
Иван поехал куда	1	куда → домой, чтобы
Иван поехал домой, чтобы зачем	0	зачем → забрать
Иван поехал домой, чтобы забрать что	1	что → одежду

Окончательно получилось следующее предложение: *Иван поехал домой, чтобы забрать одежду*.

Недостатками этого метода являются невозможность передавать большие объемы информации, а также низкая производительность метода. Кроме того, необходимо отметить невысокую скрытность секретного сообщения, которое в сильной мере влияет на структуру передаваемого текста.

Еще одним методом, близким к мимикрии, является **Spammimic** (уподобление спаму). Здесь в качестве контейнера используется обычный спам (или любой нейтральный текст, см. рис. 7.10), внутри которого размещаются установленным обеими сторонами способом значащие символы (стегосообщение).

Dear Friend, Especially for you - this red-hot intelligence. We will comply with all removal requests. This mail is being sent in compliance with Senate bill 2116, Title 9; Section 303! THIS IS NOT A GET RICH SCHEME. Why work for somebody else when you can become rich inside 57 weeks. Have you ever noticed most everyone has a cellphone & people love convenience. Well, now is your chance to capitalize on this. WE will help YOU SELL MORE and sell more! You are guaranteed to succeed because we take al the risk! But don't believe us. Ms. Simpson of Washington tried us and says "My only problem now is where to park al my cars". This offer is 100% legal. You will blame yourself forever if you don't order now! Sign up a friend and you'll get a discount of 50%. Thank-you for your serious consideration of our offer. Dear Decision maker: Thank-you for your interest in our briefing.

If you are not interested in our publications and wish to ...

Рис. 7.10. Текст-контейнер с осажденным в нем сообщением «*Здравствуйте*!»

Существует и множество других методов преобразования текста. В любом случае, при разработке эффективных лексических стеганографических методов необходимо искать золотую середину: контейнер должен быть «плотно» насыщен стегоинформацией и при этом совершенно не должен выделяться из обычной общей массы файлов такого же формата и наполнения.

В табл. 7.2 представлены результаты выполненного автором данного пособия сравнительного анализа эффективности некоторых из проанализированных методов.

Таблица 7.2

Сравнительные результаты анализа эффективности синтаксических методов текстовой стеганографии

Метод	Количество стегознаков	Плотность заполнения, %	Часть стегосообщения, содержащаяся в каждом символе контейнера, бит
Line-shift coding	811 998	1,2	0,013
Word-shift coding	8 349 980	13,1	0,132
Feature coding	49 145 754	77,6	0,776
Метод изменения регистра буквы	49 145 754	77,6	0,776
Метод изменения цвета символов	63 328 767	100,0	15,000
Метод изменения масштаба символов	63 328 767	100,0	3,000
Изменения цветовых координат[22]	63 328 767	100,0	3,000

Под «плотностью заполнения» (третий столбец таблицы) будем понимать отношение стегознаков в пустом контейнере к общему числу символов в пустом контейнере. Например, для метода изменения регистра буквы и метода *feature coding* стегознаками являются буквы любого алфавита, т. е. пробелы, цифры, специальные знаки и символы не учитываются. А в методах изменения цвета символов, изменения масштаба символов стегознаками являются все символы документа, в том числе специальные знаки и символы.

До сих пор вопрос о создании безопасной лингвистической стегосистемы остается открытым. Любая обработка текста редактором, его печать или перевод в другой формат может изменить расположение пробелов и уничтожить скрытый текст.

[22] На основе метода LSB.

7.2.2. Стеганографические методы на основе избыточности среды

Если текст-контейнер рассматривать на основе определенной цветовой модели (bitmap или RGB), то для решения наших задач могут быть адаптированы известные методы графической стеганографии. Имеется в виду то, что младшие разряды цифровых отсчетов, формирующих изображение, содержат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и дает возможность скрытия конфиденциальной информации.

К числу классических методов данного класса относится LSB (Least Significant Bit – наименее значащий бит) и различные его модификации.

Метод LSB основывается на ограниченных способностях зрения или слуха человека, вследствие чего людям тяжело различать незначительные вариации цвета или звука. Рассмотрим это на примере 24-битного растрового RGB-изображения. Каждая точка кодируется 3 байтами, каждый байт определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) цветов. Совокупность интенсивностей цвета в каждом из 3 каналов определяет оттенок пикселя.

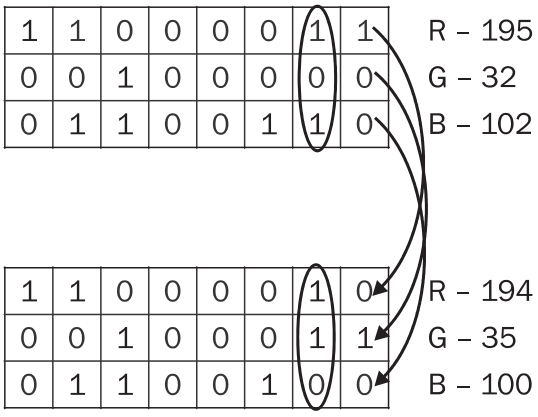


Рис. 7.11. Пример, показывающий принцип реализации метода LSB

Представим пиксель тремя байтами в битовом виде, как это показано на рис. 7.11. Младшие биты (справа) дают незначительный вклад в изображение по сравнению со старшими. Замена одного или двух младших бит для человеческого глаза будет почти незаметна.

Пример 7.3. Пусть необходимо в этом пикселе скрыть 6 битов – 101100. Разделим их на 3 пары и заменим этими парами младшие биты в каждом канале. Получили новый цвет, очень похожий на первоначальный (к сожалению, в черно-белом изображении это практически не заметно). Оценим эффективность такого метода: используя 2 бита на канал, мы сможем прятать 3 байта информации на 4 пикселя изображения. А это уже примерно 25% картинки. Таким образом, в мегабайтовом файле можно спрятать 250 Кбайт информации, причем для невооруженного глаза этот факт останется незаметен.

Недостатки метода:

1) скрытое сообщение легко разрушить, например, при сжатии или отображении;

2) не обеспечена секретность встраивания информации. Точно известно местоположение зашифрованной информации. Для преодоления этого недостатка можно встраивать информацию не во все пиксели изображения, а лишь в некоторые из них, определяемые по псевдослучайному закону в соответствии с ключом, известным только законному пользователю.

Альтернативным подходом является моделирование характеристик поведения LSB. Встраиваемое сообщение будет в этом случае частично или полностью зависеть от контейнера. Процесс моделирования является вычислительно трудоемким, кроме того, его надо повторять для каждого отдельно взятого контейнера. Главным недостатком этого метода является то, что процесс моделирования может быть повторен нарушителем, возможно обладающим большим вычислительным ресурсом, создающим лучшие модели, что приведет к обнаружению скрытого сообщения.

При использовании любого стеганографического метода на основе избыточности среды существует возможность повысить степень надежности скрытия, жертвуя при этом объемом скрываемых данных.

Объем встроенных данных и степень модификации контейнера изменяется от метода к методу. Также очевиден и тот факт, что *в зависимости от целей, для которых используется скрытие данных, различными являются и требования относительно уровня устойчивости системы к модификации контейнера*. Как следствие, для разных целей оптимальными будут разные методы стеганографии. В дополнение к табл. 7.2 в табл. 7.3 приведены сравнительные характеристики стеганографических методов в более общем плане.

Таблица 7.3

Сравнительные характеристики некоторых стеганографических методов

Стеганографический метод	Краткая характеристика метода	Недостатки метода	Достоинства метода
1. Методы использования специальных свойств компьютерных форматов данных			
Методы использования компьютерных форматов данных	Поля расширения имеются во многих мультимедийных форматах, они заполняются нулевой информацией и не учитываются программой	Низкая степень скрытности (устойчивости к взлому). Передача небольших ограниченных объемов информации	Простота использования
Методы использования известного смещения слов, предложений, абзацев	Основаны на изменении положения строк и расстановки слов в предложении, что обеспечивается вставкой дополнительных пробелов между словами	Слабая производительность метода, передача небольших объемов информации. Низкая степень скрытности	Простота использования. Имеется опубликованное программное обеспечение реализации данного метода
Методы выбора определенных позиций букв (нулевой шифр)	Акростих – частный случай этого метода (например, начальные буквы каждой строки образуют сообщение)		
Методы использования специальных свойств полей форматов, не отображаемых на экране	Основаны на использовании специальных «невидимых», скрытых полей для организации сносок и ссылок (например, использование черного шрифта на черном фоне)		
Методы скрытия в неиспользуемых местах памяти на дисках	Информация записывается в обычно неиспользуемых местах диска (например, в нулевой дорожке)		

Окончание табл. 7.3

Стеганографический метод	Краткая характеристика метода	Недостатки метода	Достоинства метода
Методы использования имитирующих функций (mimic function)	Метод основан на генерации текстов и является обобщением акростиха. Для тайного сообщения генерируется осмысленный текст, скрывающий само сообщение	Слабая производительность метода, передача небольших объемов информации. Низкая степень скрытности	Результирующий текст не является подозрительным для систем мониторинга сети
Методы удаления идентифицирующего файл заголовка	Скрываемое сообщение шифруется и у результата удаляется идентифицирующий заголовок, остаются только шифрованные данные. Получатель заранее знает о передаче сообщения и имеет недостающий заголовок	Проблема скрытия решается только частично. Необходимо заранее передать часть информации получателю	Простота реализации. Многие средства (White Noise Storm, S-Tools) обеспечивают реализацию этого метода с PGP шифроалгоритмом
2. Методы использования избыточности аудио и визуальной информации			
Методы использования избыточности цифровых фотографии, цифрового звука и цифрового видео	Младшие разряды цифровых отсчетов содержат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и дает возможность скрытия конфиденциальной информации	За счет введения дополнительной информации искажаются статистические характеристики цифровых потоков. Для снижения компрометирующих признаков требуется коррекция статистических характеристик	Возможность скрытой передачи большого объема информации. Возможность защиты авторского права, скрытого изображения товарной марки, регистрационных номеров и т. п.

7.3. Основные принципы стеганографического анализа

Определение 7.5. *Стеганоанализ* – процесс оценки перехваченного контейнера на предмет наличия в нем скрытого (осажденного) сообщения.

По сути, *стегоаналитик* и *криптоаналитик* преследуют одинаковые цели.

Основной *целью стеганоанализа* является моделирование стеганографических систем и их исследование для получения качественных и количественных оценок надежности использования стеганопреобразования, а также построение методов выявления скрываемой в контейнере информации, ее модификации или разрушения.

Терминология *стеганоанализа* аналогична терминологии *криптоанализа*, однако имеют место некоторые существенные расхождения. Как мы ранее выявили (см., например, главу 3), криптоанализ применяется с целью дешифрования криптограмм, а стеганоанализ, прежде всего, для выявления наличия скрытой информации.

Осаждение информации в электронном документе-контейнере требует изменений (перестройки) параметров или даже элементов структуры последнего. Такие изменения могут выполнить роль «следа», который сигнализирует о существовании встроенного сообщения, и, таким образом, основная идея стеганографии – скрытие факта существования секретной информации – не будет выполненной.

Стеганоанализ на предмет наличия скрытой информации может приобретать разные формы: обнаружение наличия (детектирование), извлечение и, наконец, удаление или разрушение скрытых данных. Кроме того, нарушитель может поверх уже существующей скрытой информации встроить определенную дезинформацию.

По уровню обеспечения секретности стеганографические системы делятся на теоретически устойчивые, практически устойчивые и неустойчивые.

Теоретически устойчивая стеганосистема осуществляет скрытие информации только в тех фрагментах контейнера, изменение которых не превышает уровень, определяемый случайными факторами. При этом существует теоретическое доказательство того,

что невозможно создать стеганоаналитический метод выявления скрытой информации. Здесь мы можем обнаружить некоторую схожесть со свойством криптосистемы на основе *одноразовых блокнотов*.

Практически устойчивая стеганосистема выполняет такую модификацию фрагментов документа-контейнера, которая может быть выявлена, но при этом известно, что на данный момент необходимые стеганоаналитические методы у нарушителя отсутствуют или пока что не разработаны.

Неустойчивая стеганосистема скрывает информацию таким образом, что существующие стеганоаналитические средства позволяют ее выявить. В этом случае стеганографический анализ помогает найти уязвимые места стеганографического преобразования и усовершенствовать его таким образом, чтобы все изменения, внесенные в документ-контейнер, оказались бы в области теоретической или практической неразличимости.

Определение 7.6. Стеганографическая система считается *взломанной*, если стегоаналитику (или просто нарушителю) удалось, по крайней мере, доказать существование скрытого сообщения в перехваченном контейнере.

Предполагается, что аналитик способен осуществлять любые типы атак и имеет неограниченные вычислительные возможности. Если ему не удается подтвердить гипотезу о том, что в контейнере скрыто секретное сообщение, то стеганографическая система считается *устойчивой*.

Обычно выделяют несколько *этапов взлома стеганографической системы*:

– обнаружение факта наличия скрытой информации;

– извлечение скрытого сообщения.

Возможны и иные деструктивные действия:

– модификация скрытой информации;

– блокировка дальнейшие использования скрытой информация даже на принципах санкционированного доступа.

Первые два этапа действия относят к *пассивным атакам на стеганосистему*, два других – *к активным (или злонамеренным) атакам*.

По аналогии с криптоанализом выделяют следующие общие виды атак на стеганосистемы.

Атака на основании известного заполненного контейнера. В этом случае злоумышленник имеет в своем распоряжении один или несколько заполненных контейнеров (в последнем случае предполагается, что встраивание скрытой информации выполнялось тем же самым способом). Задача нарушителя может заключаться в выявлении факта осаждения информации (наличия стеганоканала), а также в извлечении данных или определении ключа. Зная ключ, нарушитель имеет возможность анализа других стеганосообщений.

Атака на основании известного встроенного сообщения. Этот тип атаки характерен для систем защиты права интеллектуальной собственности, например, на текстовые документы или коды программ. Задачей анализа является получение ключа. Если соответствующий скрытому сообщению заполненный контейнер неизвестен, то задача является практически неразрешимой.

Атака на основании выбранного скрытого сообщения. В этом случае нарушитель может предлагать для передачи свои сообщения и анализировать полученные при этом контейнеры-результаты.

Адаптивная атака на основании выбранного сообщения. Эта атака является частным случаем предыдущей. При этом нарушитель имеет возможность выбирать сообщения для навязывания их передачи адаптивно, в зависимости от результатов анализа предшествующих контейнеров-результатов.

Атака на основании выбранного заполненного контейнера. Этот тип атаки более характерен для систем с использованием ЦВЗ. У аналитика есть некий анализатор заполненных контейнеров в виде «черного ящика» и несколько таких контейнеров. Анализируя выявленные скрытые сообщения, нарушитель пытается раскрыть ключ.

Атака на основании известного пустого контейнера. Если последний известен нарушителю, то путем сравнения его с подозреваемым на присутствие скрытых данных контейнером он всегда может установить факт наличия стеганоканала. Как видим, данный тип атаки не имеет аналога в криптографии.

Атака на основании выбранного пустого контейнера. В этом случае злоумышленник способен принудить воспользоваться предложенным им контейнером. В этом случае мы также не найдем сходства с криптографией. Как и в последнем случае.

Атака на основании известной математической модели контейнера или его части. При этом аналитик пытается определить

отличие подозреваемого сообщения от известной ему модели. Например, можно допустить, что биты в середине определенной части документа-контейнера имеют определенную взаимосвязь. Тогда отсутствие такой корреляции может служить сигналом о наличии скрытого сообщения.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. В чем состоят схожесть и основное отличие между криптографической и стеганографической системами?
2. Дайте общую классификацию и соответствующую характеристику стеганографических систем.
3. В чем состоят особенности методов текстовой стеганографии? Дайте сравнительную характеристику методов.
4. Разработайте алгоритмы практической реализации выбранного (или по указанию преподавателя) метода текстовой или графической стеганографии.
5. Охарактеризуйте известные и/или предложите собственные методы стегоанализа.
6. Классифицируйте атаки на стеганографические системы.

Глава 8
МОДЕЛИРОВАНИЕ СТЕГАНОГРАФИЧЕСКОЙ СИСТЕМЫ

Определение 8.1. Абстрактно стеганографическая система обычно определяется как некоторое *множество отображений одного пространства* (множества возможных сообщений) в другое пространство (множество возможных стегосообщений), или наоборот.

Модель будем строить на основе следующих обозначений и положений. Пусть $\mathbf{M}$ – это конечное множество сообщений, которые могут быть тайно размещены в контейнере: $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$;

$\mathbf{C}$ – это конечное множество всех допустимых контейнеров (файлов-контейнеров или документов-контейнеров): $\mathbf{C} = \{C_1, C_2, \ldots, C_p\}$, причем $p > n$.

$\mathbf{K}$ – множество всех ключей, под которыми в общем случае будем понимать методы или алгоритмы осаждения сообщения в контейнере или иные операции по предварительному преобразованию осаждаемого сообщения или выбору элементов контейнера для такого осаждения: $\mathbf{K} = \{K_1, K_2, \ldots, K_z\}$.

Произвольное тайное сообщение M_i можно скрыть в контейнере C_j при использовании ключа K_m: $M_i \in \mathbf{M}$, $i = 1, 2, \ldots, n$; $C_j \in \mathbf{C}$, $j = 1, 2, \ldots, p$; $K_m \in \mathbf{K}$, $m = 1, 2, \ldots, z$. Результатом такого типа преобразований будет заполненный контейнер (или стегосообщение) S_q, относящийся к множеству заполненных контейнеров или стегосообщений $\mathbf{S}$: $\mathbf{S} = \{S_1, S_2, \ldots, S_r\}$, $q = 1, 2, \ldots, r$.

Дальнейшие рассуждения будут строиться на базе основных понятий, которые сформулированы в виде следующих определений.

Определение 8.2. Функцию $\mathbf{F}$, определенную на $\mathbf{M} \times \mathbf{C} \times \mathbf{K}$ со значениями в $\mathbf{S}$, будем отождествлять с осаждением или встраиванием сообщения M_i из множества $\mathbf{M}$ в контейнер C_j из множества $\mathbf{C}$ на основе ключа из множества $\mathbf{K}$, предусматривающего использование соответствующего алгоритма осаждения и пространственных (геометрических или иных) параметров элементов контейнера C_j множества $\mathbf{C}$:

$$\mathbf{F}: \mathbf{M} \times \mathbf{C} \times \mathbf{K} \rightarrow \mathbf{S}. \qquad (8.1)$$

Множество отображений

$$\mathbf{F} = \{F_1, F_2, \ldots, F_l\}$$

графически можно представить в виде, показанном на рис. 8.1.

Каждое конкретное отображение F_w, где $w = 1, 2, \ldots, l$, из множества $\mathbf{F}$ соответствует конкретному алгоритму или способу осаждения информации M_i в контейнер C_j при помощи конкретного ключа K_w. Схематично представленное на рис. 8.1 отображение как раз поясняет такое взаимодействие компонент системы.

Соотношение (8.1) формально описывает процедуру осаждения сообщения в контейнере на основе выбранного метода.

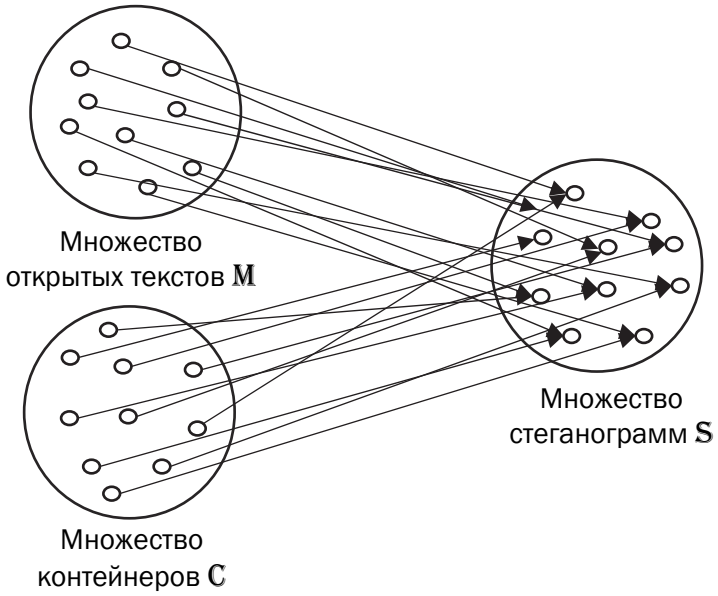


Рис. 8.1. Графические представление отображения F_w на основе ключа $K_w \in \mathbb{K}$

Определение 8.3. Функцию $\mathbb{F}^{-1}$, определенную на $\mathbb{S} \times \mathbb{K}^*$ ($\mathbb{K}^* = \{K^*_1, K^*_2, \ldots, K^*_z\}$, в общем случае $K_m \neq K^*_m$; $K_m \in \mathbb{K}$, $K^*_m \in \mathbb{K}^*$; $m = 1, 2, \ldots, z$) со значениями в $\mathbb{M}$, будем отождествлять с извлечением тайного сообщения $M_i \in \mathbb{M}$ из стегосообщения $S_q \in \mathbb{S}$:

$$\mathbb{F}^{-1}: \mathbb{S} \times \mathbb{K}^* \rightarrow \mathbb{M}, \mathbb{C}. \qquad (8.2)$$

Множеств $\mathbb{F}^{-1}$ состоит из l элементов:

$$\mathbb{F}^{-1} = \{F_1^{-1}, F_2^{-1}, \ldots, F_l^{-1}\},$$

где каждому конкретному отображению F_w ($w = 1, 2, \ldots, l$) соответствует фиксированный ключ $K^*_w \in \mathbb{K}^*$.

Таким образом, выражение (8.2) определяет обратное по отношению к (8.1) отображение, которое каждому элементу S_q множества $\mathbb{S}$ и фиксированного элемента множества $\mathbb{K}^*$ ставит в соответствие элемент M_i множества $\mathbb{M}$ и элемент C_j множества $\mathbb{C}$. Графическое представление такого отображения иллюстрирует рис. 8.2.

Соотношение (8.2) формально описывает процедуру извлечения сообщения из контейнера на основе того же выбранного метода, т. е. каждое конкретное отображение F_w^{-1}, где $w = 1, 2, \ldots, l$, из множества F^{-1} соответствует конкретному алгоритму или способу осаждения информации M_i в контейнер C_j при помощи конкретного ключа K_w^*.

Определение 8.4. ***Коллизией стеганографического преобразования*** **(или пересечением) называем ситуацию, при которой различные элементы, относящиеся к одинаковым множествам: (m_a, c_a, k_a) и (m_b, c_b, k_b) – формируют в результате одинаковые отображения, что формально можно записать в следующем виде:**

$$(m_a, c_a, k_a) = (m_b, c_b, k_b),$$

причем $m_a \neq m_b$, $c_a \neq c_b$, $k_a \neq k_b$, либо неравенство выполняется хотя бы для одной из трех пар и $1 \leq a, b \leq n, p, z$.

Определение 8.5. ***Стеганографической системой*** $\sum$ **будем называть совокупность сообщений $\mathbb{M}$, контейнеров $\mathbb{C}$, ключей $\mathbb{K}$, стегосообщений (заполненных контейнеров) $\mathbb{S}$ и преобразований (прямого $\mathbb{F}$ и обратного $\mathbb{F}^{-1}$), которые их связывают:**

$$\sum = (\mathbb{M}, \mathbb{C}, \mathbb{K}, \mathbb{S}, \mathbb{F}, \mathbb{F}^{-1}). \qquad (8.3)$$

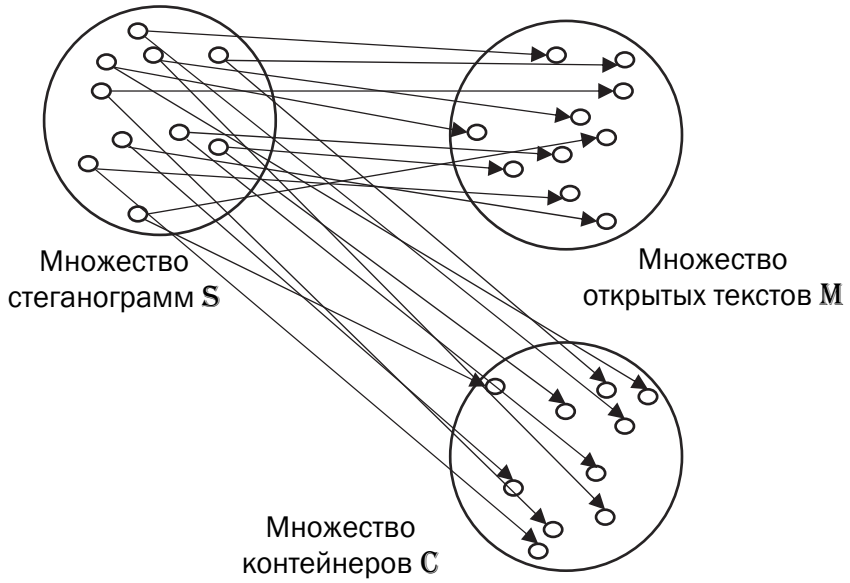


Рис. 8.2. Графические представление отображения F_w^{-1} на основе ключа $K_w^* \in \mathbb{K}^*$

Вернемся к обсуждению множества ключевой информации $\mathbb{K}$. Формально этот тип множества, входящего в состав системы $\sum$, можно рассматривать состоящим из некоторого числа непересекающихся подмножеств. Упомянутое число зависит от того, сколь-

ко факторов влияют на стойкость к взлому созданной системы. Под такими факторами (следуя постулату Керкгоффса) будем понимать ключевую информацию. Иначе говоря, количество ключей.

Если тайна осаждения информации в контейнер обусловлена лишь алгоритмом или способом реализации такой операции, то можно говорить лишь о простом множестве ключей K, т. е не содержащем подмножеств. Единственным ключом в подобных системах является собственно сам алгоритм или способ осаждения/извлечения информации[23]. И для таких стеганографических систем справедливо формальное их описание в виде выражения (8.3).

Однако для дополнительной защиты от взлома или иного влияния на осажденную информацию, как правило, применяют дополнительные меры. Такими мерами могут быть:

а) симметричные или асимметричные ключи для зашифрования/расшифрования информации, относящейся к множеству $\mathbb{M}$, при ее осаждении/извлечении;

б) алгоритмы или методы помехоустойчивого кодирования информации M_i ($M_i \in \mathbb{M}$) с целью обнаружения и/или исправления ошибок в этой информации, появившихся в процессе передачи информации или ее хранения в этом контейнере, и другие подобные.

Каждый из рассмотренных выше случаев может быть отождествлен с использованием дополнительного (к базовому алгоритму осаждения/извлечения M_i) ключа.

Таким образом, мы можем говорить о *классе стеганосистем с дополнительным ключом*. Каждый такой ключ $K_m^{д}$ относится к подмножеству дополнительных ключей $\mathbb{K}^{д}$.

Определение 8.6. ***Дополнительным ключом*** $\mathbb{K}^{д}$ **стеганографической системы будем считать конкретное секретное значение набора параметров криптографического или иного алгоритма, используемое для криптографического зашифрования/расшифрования сообщения, для помехоустойчивого кодирования/декодирования сообщения или иной дополнительной операции, используемой при осаждении/извлечении сообщения** M_i; $K^{A} = \{K_1^{A}, K_2^{A}, ..., K_t^{A}\}$ **в качестве дополнительного средства повышения стеганографической стойкости системы.**

[23] Хотя, например, Конахович Г. Ф., Пузыренко А. Ю. (Компьютерная стеганография. Теория и практика. Киев: МК-Пресс, 2006. 288 с.) такие стеганосистемы относят к классу *бесключевых*.

Как отдельный тип ключа можно рассматривать псевдослучайные или иные метки, определяющие позиции или местоположение элементов контейнера, которые подвергаются модификации при размещении в этом контейнере определенного элемента (символа осаждаемого сообщения M_i) (такие метки, например, могут представлять собой позицию пикселя в матрице, формирующей изображение, текст или отдельный символ документа-контейнера при модификации его цветовой, яркостной или иной характеристики в процессе осаждения/извлечения тайного сообщения).

Значение такой информации в обеспечении общей стойкости стеганографической системы к взлому сопоставимо со значением дополнительного ключа, задаваемого из множества $\mathbf{K}^{д}$. Однако исходя из физических принципов функционирования стеганосистем, ключевую информацию, относящую к принципам выбора местоположения для размещения конкретной единицы информации (бита, байта, символа и т. д.) сообщения $M_i \in \mathbf{M}$, $i = 1, 2, \ldots, n$ в документе-контейнере $C_j \in \mathbf{C}$, $j = 1, 2, \ldots, p$, можно относить к множеству $\mathbf{K}$.

Поскольку генерация и использование дополнительных ключей практически не связаны с базовым алгоритмом осаждения/извлечения сообщения M_i в контейнере C_j с помощью ключа K_m, целесообразно говорить о *двухключевых стеганографических системах*.

Определение 8.7. ***Двухключевой стеганографической системой*** $\sum_2$ **будем называть совокупность сообщений** $\mathbf{M}$**, контейнеров** $\mathbf{C}$**, ключей** $\mathbf{K}$**, дополнительных ключей** $\mathbf{K}^{д}$**, стегосообщений (заполненных контейнеров)** $\mathbf{S}$ **и преобразований (прямого** $\mathbf{F}$ **и обратного** $\mathbf{F}^{-1}$**), которые их связывают:**

$$\sum_2 = (\mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{K}^{д}, \mathbf{S}, \mathbf{F}, \mathbf{F}^{-1}). \quad (8.4)$$

В соответствии с выражением (8.4) стеганографические преобразования (осаждение и извлечение информации) для двухключевых систем в общем виде описываются соотношениями:

$$\mathbf{F}: \mathbf{M}\times\mathbf{C}\times\mathbf{K}\times\mathbf{K}^{д} \rightarrow \mathbf{S}, \quad (8.5)$$

$$\mathbf{F}^{-1}: \mathbf{S}\times\mathbf{K}\times\mathbf{K}^{д} \rightarrow \mathbf{M}, \mathbf{C}. \quad (8.6)$$

Отображения, задаваемые выражениями (8.1), (8.2), (8.5), (8.6), следует относить к числу *функциональных*, поскольку между каждым элементом множеств, указанных в левой части (от стрелки) и в

правой части, должно существовать однозначное соответствие. В дополнение к этому отметим, что в стеганографичеких системах, основным назначением которых является передача информации и состояние (информативность) контейнера значения не имеет, выражения (8.2) и (8.6) могут быть записаны в каноническом виде:

$$\mathbf{F}^{-1}\colon \mathbf{S}\times\mathbf{K} \to \mathbf{M}, \tag{8.7}$$

$$\mathbf{F}^{-1}\colon \mathbf{S}\times\mathbf{K}\times\mathbf{K}^{д} \to \mathbf{M}. \tag{8.8}$$

Однако для систем, относящихся к задачам охраны прав интеллектуальной собственности (на документ-контейнер), единственно справедливыми следует рассматривать выражения (8.2) и (8.6). Это следует из логики самой задачи: обратное преобразование должно в одинаковой и полной мере восстанавливать исходные элементы системы: тайную (авторскую) информацию $M_i \in \mathbf{M}$ и документ-контейнер $C_j \in \mathbf{C}$, авторство которого должно быть подтверждено с помощью M_i.

Теперь обратимся к множеству $\mathbf{K}^{д}$. В силу рассуждений, изложенных выше, это множество представим в виде непересекающихся подмножеств, число которых соответствует количеству типов используемых в системе дополнительных ключей (например, криптографических – им соответствует подмножество $\mathbf{K}^{дк}$).

Если дополнительные ключи ограничиваются только криптографическими ключами, то указанное множество $\mathbf{K}^{д}$ и подмножество $\mathbf{K}^{дк}$ совпадают.

Зафиксировав множество всех ключей $\mathbf{K} = \{K_1, K_2, \ldots, K_z\}$ и множество всех дополнительных ключей $\mathbf{K}^{д}$ в виде криптографических ключей $\mathbf{K}^{дк} = \{K_1^{дк}, K_2^{дк}, \ldots, K_t^{дк}\}$ так, чтобы для всех $w = 1, 2, \ldots, l$ отображение F_w в соответствии с выражением (8.5) однозначно задавалось этими ключами, т. е. $F_w \in \mathbf{F}$ (и соответственно для $(F^{-1})_w \in \mathbf{F}^{-1}$ – для (8.6)), можем формально переписать это в следующем виде для одного фиксированного набора ключей – $K_w^{дк}$ и K_w:

$$F_w\colon M_i \xrightarrow{K_w^{дк}} M_i'; \qquad M_i' \times C_j \xrightarrow{K_w} S_w \tag{8.9}$$

и

$$(F^{-1})_w\colon S_w \xrightarrow{K_w} M_i', C_j; \qquad M_i' \xrightarrow{K_w^{дк}} M_i, \tag{8.10}$$

где $w = 1, 2, \ldots, l$.

В последнем случае каждое из преобразований – (8.9) или (8.10) – можно рассматривать как аддитивный процесс. Тогда гра-

фическое представление соответствующих отображений, например (8.9), строится с учетом того, что первая часть $F_{w(1)}$ отображения есть не что иное, как отображение множества открытых сообщений в множество криптограмм или шифрограмм на основе, например, ключа зашифрования/расшифрования $K_w^{\text{дк}}$ (рис. 8.3).

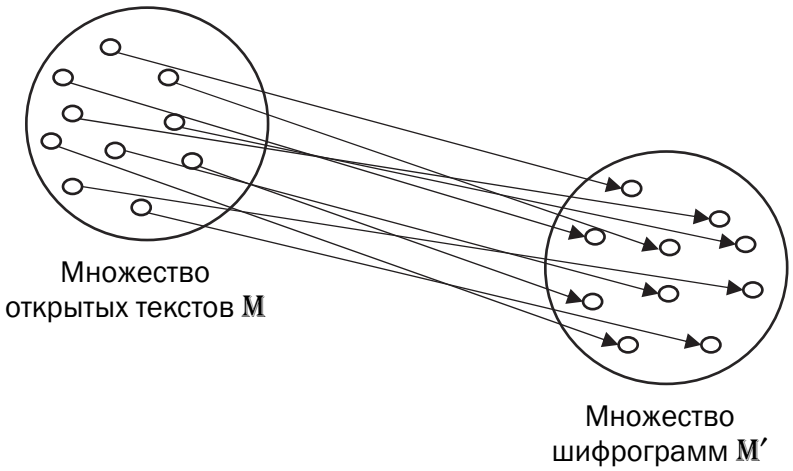


Рис. 8.3. Графические представление первой части отображения в двухключевой системе на основе выражения (8.9):

$$F_{w(1)}: M_i \xrightarrow{K_w^{\text{дк}}} M_i'$$

Вторая часть отображения F_w ($F_{w(2)}$) на основе выражения (8.9) графически будет выглядеть так же, как на рис. 8.1, с той лишь разницей, что вместо элементов множества открытых текстов $\mathbf{M}$ следует использовать множество $\mathbf{M}'$:

$$F_{w(2)}: M_i' \times C_j \xrightarrow{K_w} S_w.$$

Следует помнить о важном условии однозначности обратных преобразований: *мощности связанных между собой ключами преобразований множеств* $\mathbf{M}$ и $\mathbf{S}$ *должны быть одинаковыми*. Если прямое и обратное криптографическое преобразования осуществляются на основе асимметричной системы, то ключевая информация при соответствующем прямом и обратном стеганографическом преобразовании будет разной. Это мы подробно рассмотрели в подглаве 5.6. Дополнительно мы это обстоятельство предусмотрели, приняв $K_m \neq K_m^*$; $K_m \in \mathbf{K}$, $K_m^* \in \mathbf{K}^*$; $m = 1, 2, \ldots, z$, где в общем случае K_m и K_m^* – фиксированные значения ключа прямого и обратного преобразований соответственно.

Для подобных систем выражения (8.9) и (8.10), строго говоря, должны быть изменены:

$$F_w: M_i \xrightarrow{(K_w^{\text{дк}})_\text{з}} M_i'; \quad M_i' \times C_j \xrightarrow{K_w} S_w \tag{8.11}$$

и

$$F_w^{-1}: S_w \xrightarrow{K_w} M_i', C_j; \quad M_i' \xrightarrow{(K_w^{\text{дк}})_\text{р}} M_i, \tag{8.12}$$

поскольку ключ зашифрования не равен ключу расшифрования: $(K_w{}^{\text{дк}})_\text{з} \neq (K_w{}^{\text{дк}})_\text{р}$; $K_w{}^{\text{дк}} = ((K_w{}^{\text{дк}})_\text{з},(K_w{}^{\text{дк}})_\text{р})$ при операциях над выбранным контейнером и сообщением.

Далее обсудим вопрос о последовательности использования элементов ключевых множеств при осаждении и извлечении информации в контейнере.

Как следует из выражений (8.9) и (8.11), а также исходя из физических особенностей рассматриваемых процессов, дополнительный ключ $K_w{}^{\text{дк}}$ (равно как и $K_w{}^{\text{д}}$) из множества ключей $\mathbf{K}^{\text{дк}}$ ($\mathbf{K}^{\text{д}}$) при осаждении информации используется перед основным процессом – осаждения, а при обратной операции – после извлечения сообщения из контейнера. В случае же использования *составного дополнительного ключа* (например, шифрование и помехоустойчивое кодирование сообщения M_i), состоящего в общем случае из v подключей, относящихся к соответствующим элементам подмножества $\mathbf{K}^{\text{д}}$: $\mathbf{K}^{\text{д}} = \{K_1{}^{\text{д}}, K_2{}^{\text{д}}, \ldots, K_v{}^{\text{д}}\}$; $\boldsymbol{K}_1{}^{\text{д}} = \{K_{11}{}^{\text{д}}, K_{12}{}^{\text{д}}, \ldots, K_{1n}{}^{\text{д}}\}$, …, $\boldsymbol{K}_v{}^{\text{д}} = \{K_{v1}{}^{\text{д}}, K_{v2}{}^{\text{д}}, \ldots, K_{vm}{}^{\text{д}}\}$, последовательность использования такого подключа при извлечении информации будет обратной (или зеркальной) по отношению к процессу осаждения.

Обратимся сейчас к ключу, образованному множеством $\mathbf{K}$. Мы говорили о том, что в простейшем случае к ключевой информации можно отнести метод или способ осаждения. Для *стеганографических систем передачи информации* может быть выбран один из возможных типов контейнеров (текст, графика, аудио и т. д.), для каждого из которых может быть реализован один из возможных алгоритмов или способов осаждения (например, для текстового документа-контейнера – метод *Line-shift coding*, *Word-shift coding* или иной). В задачах охраны права интеллектуальной собственности на конкретный электронный документ имеет значение лишь алгоритм осаждения.

Таким образом, одно подмножество ключей ($\mathbf{K}^{\mathbf{1}}$) множества ключей $\mathbf{K}$ образуют типы контейнеров ($\mathbf{K}^{\mathbf{1}\boldsymbol{c}}$) и алгоритмы (способы) осаждения информации $M_i \in \mathbf{M}$ ($\mathbf{K}^{\mathbf{1}\boldsymbol{a}}$): $\mathbf{K}^{\mathbf{1}} = \{\mathbf{K}^{\mathbf{1}\boldsymbol{c}}, \mathbf{K}^{\mathbf{1}\boldsymbol{a}}\}$; $\mathbf{K}^{\mathbf{1}\boldsymbol{c}} = \{K^{1c}{}_x\}$, $x = 1, 2, \ldots, c$; $\mathbf{K}^{\mathbf{1}\boldsymbol{a}} = \{K^{1a}{}_o\}$, $o = 1, 2, \ldots, h$. Другое подмножество

ключей ($\mathbf{K^2}$) множества $\mathbf{K}$ образуют указанные выше метки, определяющие в общем случае выбор элементов контейнера для осаждения соответствующих элементов сообщения. Понятно, что такие ключи также могут зависеть от типа используемого контейнера и используемого алгоритма осаждения. Для упрощения примем, что $\mathbf{K^2} = \{K^2_1, K^2_1, \ldots, K^2_d\}$.

Понятно, что ключи $\mathbf{K^{1c}}$ и $\mathbf{K^{1a}}$ могут нас интересовать в качестве самостоятельных параметров только при анализе (или вероятностной оценке) устойчивости системы к взлому, поскольку выбор типа контейнера носит не случайный, а детерминированный характер при решении задач защиты авторского права (в частности). Поэтому при формальном описании рассматриваемых процессов будем использовать только объединенный ключ, относящийся к множеству $\mathbf{K}$.

С учетом изложенного выражения (8.9) и (8.10) в общем случае примут вид:

$$\left.\begin{array}{l} F_w\colon M_i' \xrightarrow{K^{д}_{1w}} M_{i1}';\ \ M_{i1}' \xrightarrow{K^{д}_{2w}} M_{i2}';\ldots; \\ M_{i(v-1)}' \xrightarrow{K^{д}_{yw}} M_{iv}';\ \ M_{iv}' \times C_j \xrightarrow{K_w} S_w; \end{array}\right\} \qquad (8.13)$$

и

$$\left.\begin{array}{l} F^{-1}{}_w\colon S_w \xrightarrow{K_w} M_{iv}', C_j;\ \ M_{iv}' \xrightarrow{(K^{д}_{vw})^*} M_{i(v-1)}';\ldots; \\ M_{i2}' \xrightarrow{(K^{д}_{2w})^*} M_{i1}';\ \ M_{i1}' \xrightarrow{(K^{д}_{1w})^*} M_i', \end{array}\right\} \qquad (8.14)$$

где $K_w = \{K_w{}^1, K_w{}^2\}$; $K^1{}_w \in \mathbf{K^1}$, $K_w{}^2 \in \mathbf{K^2}$; $\mathbf{K} = \{\mathbf{K^1}, \mathbf{K^2}\}$; $\mathbf{K^д} = \{\boldsymbol{K}_1{}^{д}, \boldsymbol{K}_2{}^{д}, \ldots, \boldsymbol{K}_v{}^{д}\}$ и $K_{1w}{}^{д} \in \mathbf{K}_1{}^{д}$, $K_{2w}{}^{д} \in \mathbf{K}_2{}^{д}$; ….; $K_{vw}{}^{д} \in \boldsymbol{K}_v{}^{д}$; $\mathbf{K^{д*}} = \{\boldsymbol{K}_1{}^{д*}, \boldsymbol{K}_2{}^{д*}, \ldots, \boldsymbol{K}_v{}^{д*}\}$ и $(K_{1w}{}^{д})^* \in \mathbf{K}_1{}^{д*}$, $(K_{2w}{}^{д})^* \in \mathbf{K}_2{}^{д*}$; ….; $(K_{vw}{}^{д})^* \in \boldsymbol{K}_v{}^{д*}$; $F_w \in \boldsymbol{F}$; $F_w{}^{-1} \in \boldsymbol{F^{-1}}$.

Перепишем выражения (8.13) и (8.14) в общем виде, соответствующем выражения (8.5) и (8.6), однако учитывающем последовательность операций:

$$\boldsymbol{F}\colon \mathbf{M} \times \mathbf{K^д} \times \mathbf{C} \times \mathbf{K} \rightarrow \mathbf{S}, \qquad (8.15)$$

$$\boldsymbol{F^{-1}}\colon \mathbf{S} \times \mathbf{K} \times \mathbf{K^{д*}} \rightarrow \mathbf{M}, \mathbf{C}. \qquad (8.16)$$

С учетом формальной разницы между $\mathbf{K^д}$ и $\mathbf{K^{д*}}$ выражение (8.4) окончательно запишем так:

$$\sum\nolimits_2 = (\mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{K^д}, \mathbf{K^{д*}}, \mathbf{S}, \mathbf{F}, \mathbf{F^{-1}}). \qquad (8.17)$$

В соответствии с формулой (8.15) выбранные ключи из одного множества дополнительных ключей ($\mathbf{K}^{д}$) используются для предварительного одно- или многократного (v-кратного) преобразования осаждаемого в контейнер $C_j \in \mathbf{C}$ сообщения $M_i \in \mathbf{M}$, а выбранные ключи из другого множества ключей ($\mathbf{K}$) используются непосредственно при реализации операции осаждения. Соответственно ключи, относящиеся ко множеству $\mathbf{K}^{д*}$, используются при обратном v-кратном преобразовании на приемной стороне стеганографической системы, описываемой выражением (8.17).

Таким образом, выражения (8.13) и (8.15) позволяют выполнить операцию синтеза передающей части стеганографической системы, которую по определению 8.6 мы назвали *двухключевой*. Эта часть системы будет состоять из следующих блоков: *источник потока сообщений* $M_i \in \mathbf{M}$, который формирует определенное сообщение M_i; *источник потока* (*множества*) *пустых контейнеров* (*документов-контейнеров*) $\mathbf{C}$, формирующий определенный контейнер $C_j \in \mathbf{C}$ для осаждения в нем сообщения M_i; *блок предварительного преобразования* (шифрование, кодирование и т. д.) сообщения M_i; *источник дополнительных ключей*, формирующий v ключей из множества $\mathbf{K}^{д}$: $\mathbf{K}^{д} = \{\boldsymbol{K}_1^{д}, \boldsymbol{K}_2^{д}, \ldots, \boldsymbol{K}_v^{д}\}$; $\boldsymbol{K}_1^{д} = \{K_{11}^{д}, K_{12}^{д}, \ldots, K_1^{д}\}$, …, $\boldsymbol{K}_v^{д} = \{K_{v1}^{д}, K_{v2}^{д}, \ldots, K_{vm}^{д}\}$; источник ключей $\mathbf{K}$; *блок осаждения сообщения* M_i в контейнере C_j и формирования стегосообщения S_w на основе ключа $K_w \in \mathbf{K}$, $w = 1, 2, \ldots, l$; $S_w \in \mathbf{S} = \{S_1, S_2, \ldots, S_r\}$.

Точно так же на основе соотношений (8.14) и (8.16) можем синтезировать приемную часть стеганографической системы (8.17). При этом необходимо принять во внимание обратный порядок (по отношению к вышеописанному) использования ключевой информации и формальную разницу в значениях дополнительных ключей ($\mathbf{K}^{д*}$) в сравнении с $\mathbf{K}^{д}$.

В общем случае, как и в криптографических системах, передающую и приемную часть стеганографической системы связывает канал передачи (хранения) стегосообщения. В контексте задачи построения структурной схемы стеганографической системы сейчас ограничимся тем, что представим результат влияния различных факторов (действие злоумышленника; влияние шумов; изменение формата документа-контейнера: например, изменение шрифта текстового документа-контейнера и др.), которые приводят к изменению стегосообщения $S_w \in \mathbf{S}$, в виде параметра ΔS_w.

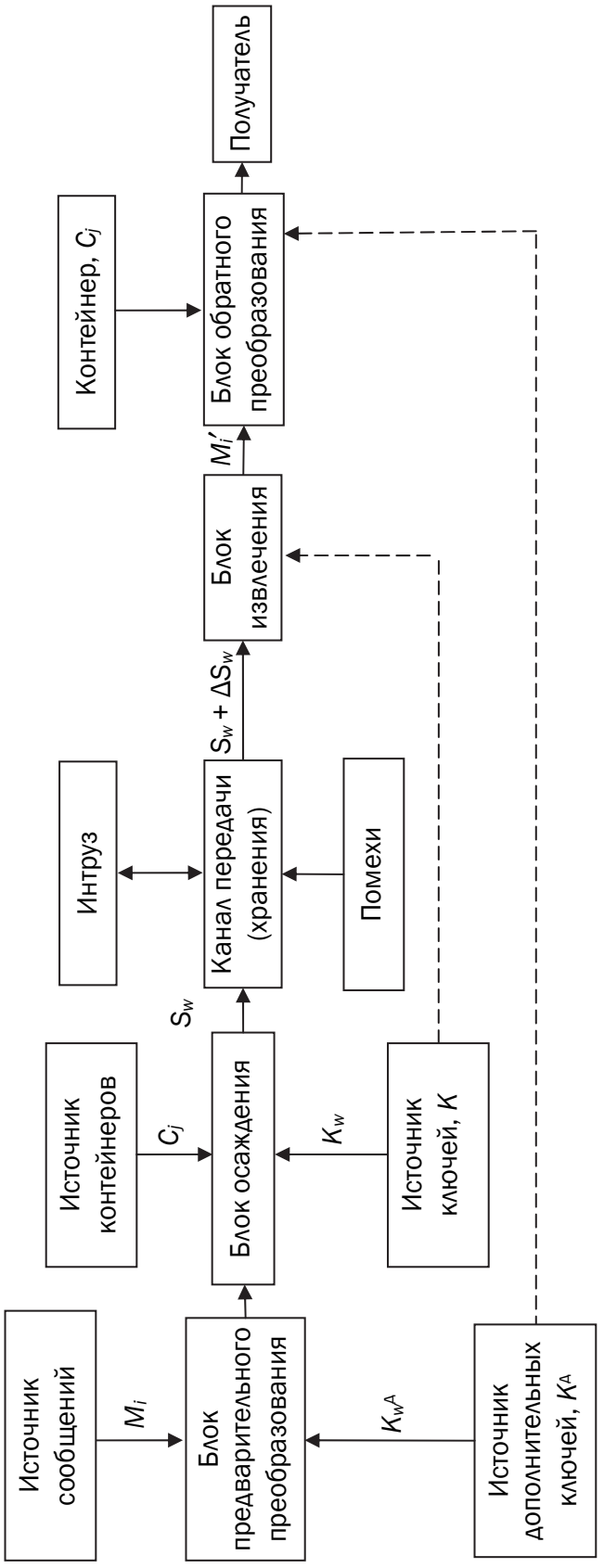


Рис. 8.4. Структурная схема двухключевой стеганографической системы

Структурная схема синтезированной двухключевой стеганографической системы, удовлетворяющей выражением (8.4), (8.13), (8.15), приведена на рис. 8.4.

В этом случае выражение (8.16) следует представить в виде

$$\boldsymbol{F}^{-1}\mathbf{:}\ (\mathbf{S} + \Delta\mathbf{S})\times\mathbf{K}\times\mathbf{K}^{д^*} \rightarrow \mathbf{M}, \mathbf{C}. \qquad (8.18)$$

Последнее будет справедливо для *робастных стеганографических систем*, т. е. для стегосистем, устойчивых к непреднамеренным воздействиям на элементы этих систем.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Представьте и поясните структурную схему стеганографической системы: а) одноключевой; б) двухключевой.

2. Дайте определения и формально опишите стеганографические системы: а) одноключевую; б) двухключевую.

Раздел IV

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Глава 9
ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ НА ОСНОВЕ СИММЕТРИЧНОЙ КРИПТОГРАФИИ

Электронная цифровая подпись (ЭЦП) является элементом криптографического преобразования информации. Однако учитывая специфическую роль этого компонента информационных систем в общем комплексе задач, которые решаются в предметной области, автор посчитал целесообразным рассмотреть некоторые важные вопросы создания (генерации) и использования ЭЦП в отдельной главе.

9.1. Назначение, структура, особенности построения и использования ЭЦП

Нередко мы в конце письма или документа ставим свою подпись. Подобное действие может преследовать две цели. Во-первых, получатель корреспонденции имеет возможность убедиться в истинности письма, сличив подпись с имеющимся у него образцом. Во-вторых, личная подпись является юридическим гарантом авторства документа. Последний аспект особенно важен при заключении торговых сделок, составлении доверенностей, обязательств и т. д.

С внедрением электронного документооборота (в том числе и конфиденциального) особо актуальной стала проблема установления подлинности и авторства безбумажной документации.

При всех преимуществах современных криптосистем они не позволяют обеспечить аутентификацию данных. Поэтому средства аутентификации должны использоваться в комплексе с криптографическими алгоритмами. Электронная цифровая подпись (ЭЦП) как раз и является тем инструментом, который призван решать отмеченные задачи.

Понятие «электронная цифровая подпись» было введено в 1976 году У. Диффи и М. Хеллманом, высказавшими предположение о ее существовании. В 1977 году Р. Ривест, А. Шамир и Л. Адлеман разработали криптосистему RSA, которую выше мы достаточно подробно проанализировали и которую можно было использовать для создания примитивных ЭЦП. Вскоре после RSA были разработаны алгоритмы цифровой подписи И. Рабина и Р. Меркле. В 1984 году Ш. Гольдвассер, С. Микали и Р. Ривест сформулировали требования безопасности к алгоритмам ЭЦП, описали атаки на ЭЦП.

ЭЦП позволяет выполнять те же функции, что и собственноручная (поставленная «от руки») подпись:

• *аутентифицировать лицо*, подписавшее сообщение; ЭЦП получается в результате криптографического преобразования электронных данных документа с использованием личного ключа ЭЦП;

• *контролировать целостность* сообщения; ЭЦП вычислена на основании исходного состояния документа и соответствует лишь ему, поэтому при любом случайном или преднамеренном изменении документа подпись станет недействительной;

• *защищать сообщение* от подделок; любая подделка должна быть выявлена путем операций сравнения соответствующих атрибутов подписанного и полученного адресатом сообщений;

• *доказать авторство* лица, подписавшего сообщение; создать корректную ЭЦП можно, лишь зная закрытый ключ, известный только его владельцу (лицу, подписавшему документ).

Важнейшие отличительные особенности ЭЦП:

• ЭЦП представляет собой бинарную последовательность (в отличие от графического образа, каковым является подпись от руки);

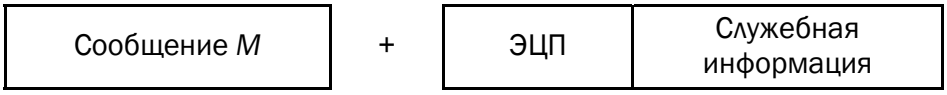
• указанная бинарная последовательность зависит от содержания подписываемого сообщения.

Определение. *Электронная цифровая подпись* – последовательность символов, являющаяся реквизитом электронного документа, зависящая от содержания этого документа и предназначенная для подтверждения целостности и подлинности электронного документа.

Существует несколько схем построения ЭЦП:

- на основе алгоритмов симметричного шифрования; авторизацией документа является сам факт зашифрования его секретным ключом;
- на основе алгоритмов асимметричного шифрования; упоминавшаяся выше в данном разделе ЭЦП на основе RSA относится именно к этому классу;
- на основе алгоритмов асимметричного шифрования и хеш-функции; это наиболее распространенная схема.

Общая структура подписанного электронного документа *M* представляет собой, как правило, конкатенацию этого документа и ЭЦП. Кроме этих двух элементов, интегральный документ может содержать некоторую служебную информацию (дата, время отправки или различные данные об отправителе), как это схематично показано на рисунке.



Структура документа с ЭЦП

Важное свойство цифровой подписи заключается в том, что ее может проверить каждый, кто имеет доступ к *открытому ключу* ее автора (здесь речь идет об ЭЦП на основе алгоритмов асимметричного шифрования). Один из участников обмена сообщениями после проверки подлинности цифровой подписи может передать подписанное сообщение еще кому-то, кто тоже в состоянии проверить эту подпись. Например, сторона *A* может переслать стороне *B* электронный чек. После того как сторона *B* проверит подпись стороны *A* на чеке, она может передать его в свой банк, служащие которого также имеют возможность проверить подпись и осуществить соответствующую денежную операцию.

И, наконец, еще одна особенность. Обычно подписываемые с помощью ЭЦП сообщения не шифруются. Они пересылаются в

исходном виде. Их содержимое не защищено от нарушения конфиденциальности. Путем совместного применения известных алгоритмов шифрования и ЭЦП можно создавать сообщения, которые будут и зашифрованы, и подписаны. Для этого автор сначала должен добавить к сообщению свою сгенерированную ЭЦП, а затем – зашифровать получившуюся в результате пару (состоящую из самого сообщения и подписи к нему) соответствующим ключом. Получатель расшифровывает полученное сообщение тем же (при симметричной системе) или публичным ключом отправителя (при использовании асимметричной системы). Если проводить аналогию с пересылкой обычных бумажных документов, то этот процесс похож на то, как если бы автор документа поставил под ним свою печать, а затем положил его в бумажный конверт и запечатал, с тем чтобы конверт был распечатан только тем человеком, кому адресовано сообщение.

9.2. ЭЦП на основе алгоритмов симметричного шифрования

Понятно, что *симметричная схема цифровой подписи* использует один и тот же ключ для генерации ЭЦП и ее проверки.

Основные функции ЭЦП, с формальной точки зрения, реализуются примитивной процедурой шифрования/расшифрования. Ведь *аутентичнось*, *целостность*, *защиту от подделок* и *доказательство авторства* обеспечивает собственно симметричный ключ, используемый двумя сторонами в процессе обмена сообщениями. Этот ключ согласован сторонами, известен (должен быть известен) только этим абонентам. Хотя в дополнение к обычным образом зашифрованному сообщению не формируется ЭЦП как самостоятельный элемент послания (см. рисунок) или просто шифртекста, как видим, формально другая сторона получает вместе с сообщением все необходимые вышеперечисленные доказательства.

Первыми, кто обратил внимание на возможность симметричной схемы ЭЦП, были основоположники самого понятия: У. Диффи и М. Хеллман, которые опубликовали описание алгоритма подписи одного бита с помощью блочного шифра. Из этого следует, что подписывать (шифровать) следует каждый бит сообщения M, т. е. по размеру ЭЦП может превосходить подписываемый до-

кумент на несколько порядков. В силу этого серьезного недостатка идея не нашла практического применения.

Однако если предусмотреть наличие в системе третьего лица – арбитра или посредника ($П$), пользующегося доверием обеих сторон, то можно избежать указанного недостатка.

Основной алгоритм состоит в следующем.

1. Посредник $П$ вырабатывает для A и B разные (сеансовые, например) ключи: K_A и K_B.

2. Абонент A шифрует свое сообщение M ключом K_A и отсылает его посреднику: $C = E_{K_A}(M)$.

3. $П$ расшифровывает C ключом K_A ($M = D_{K_A}(C)$), тем самым извлекает сообщение M. Присоединяет к этому сообщению подтверждение того, что автором его является абонент А (обозначим эту часть нового сообщения $M_Д$). Таким образом сформирован конкатенированный документ $M' = M \,||\, M_Д$. Посредник шифрует M' ключом K_B: $C_1 = E_{K_B}(M')$. Зашифрованное сообщение C_1 отсылается абоненту B.

4. Абонент B расшифровывает C_1 ключом K_B ($M' = D_{K_B}(C_1)$), тем самым извлекает сообщение M'. Из этого сообщения он получает оригинал сообщения M и подтверждение тому, что автором этого сообщения является абонент A.

Таким образом, здесь выполнены все функциональные требования, присущие ЭЦП:

1) подпись достоверна ($П$ – гарант);

2) подпись неподдельна, так как только A и B знают ключи, использовавшиеся в процедурах (к $П$ – абсолютное доверие);

3) подписанный документ нельзя изменить или подделать;

4) подпись нельзя отрицать.

Если ключи действительно были сеансовыми, то подпись невозможно использовать повторно.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Дайте определение электронной цифровой подписи.
2. Основные функции электронной цифровой подписи.
3. Общие и отличительные признаки собственноручной и электронной подписи.
4. Опишите алгоритм ЭЦП с посредником. В чем состоят основные особенности алгоритма?

Глава 10
ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ НА ОСНОВЕ АСИММЕТРИЧНОЙ КРИПТОГРАФИИ

10.1. Особенности ЭЦП на основе алгоритмов асимметричной криптографии

Асимметричные схемы ЭЦП относятся к криптосистемам с открытым ключом. Однако в отличие от асимметричных алгоритмов шифрования, в которых зашифрование производится с помощью открытого ключа получателя, а расшифрование – с помощью закрытого ключа получателя, в схемах цифровой подписи подписывание производится с применением закрытого ключа отправителя, а проверка – с применением открытого ключа отправителя.

Таким образом, следует запомнить: **асимметричная криптография (передача зашифрованных сообщений) основана на использовании ключей получателя, а асимметричная схема ЭЦП – на использовании ключей отправителя**.

При этом стойкость ЭЦП к подделыванию (криптостойкость) определяется теми же факторами, что и криптостойкость алгоритмов зашифрования/расшифрования сообщений: чтобы применение ЭЦП имело смысл, необходимо, чтобы вычисление легитимной подписи без знания закрытого ключа было вычислительно сложным процессом. Обеспечение этого во всех асимметричных алгоритмах цифровой подписи опирается на известные нам вычислительные задачи:

- *дискретного логарифмирования*;
- *факторизации*, т. е. разложения числа на простые множители.

Для ускорения процесса генерации подписи, т. е. для уменьшения объема вычислительных операций, обычно принято подписывать не само сообщение M, а его образ, который получают путем вычисления хеша (*хеширования*; иногда пишут *хэширования*) или *хеш-функции* от сообщения M: $h(M)$. Далее рассмотрим некоторые из известных алгоритмов хеширования.

10.2. Основные понятия из области хеширования сообщений

Определение 10.1. ***Хеширование*** **(англ.** ***hashing*****) – это преобразование входного массива данных определенного типа и произвольной длины в выходную битовую строку фиксированной длины. Такие преобразования также называются** ***хеш-функциями*** **или** ***функциями свертки*****, а их результаты называют** ***хешем*** **или** ***дайджестом*** **сообщения (англ.** ***message digest*****).**

Существует и иное, более общее определение.

Определение 10.2. ***Хеш-функция*** **– математическая или иная функция, которая принимает на входе строку символов переменной (произвольной) длины и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины, называемой** ***значением хеш-функции*****.**

Хеш-функцию можно использовать для преобразования произвольного входного текста в подходящий формат. Стоит также заметить, что использование хеш-функции не обязательно при вычислении ЭЦП, а сама функция не является частью алгоритма ЭП, поэтому хеш-функция может использоваться любая или не использоваться вообще.

Принято считать, что хорошей, с точки зрения практического применения, является такая хеш-функция, которая удовлетворяет следующим основным условиям:

- функция должна быть простой с вычислительной точки зрения;
- функция должна минимизировать число *коллизий*, т. е. ситуаций, когда разным сообщениям соответствует одно значение хеш-функции.

При этом первое свойство хорошей хеш-функции зависит, в основном, от параметров компьютера, а второе – от значений данных и алгоритма хеширования. В дополнение к приведенным свойствам добавим, пожалуй, важнейшее: свойство *однонаправленности*.

Определение 10.3. ***Однонаправленная*** **(или** ***односторонняя*****)** ***хеш-функция*** **предполагает простоту ее вычисления (вычисления** $h(x)$ **по известному аргументу** x**) и сложность обратного вычисления (вычисления** x **по известному** $h(x)$**).**

Как мы ранее отмечали, однонаправленность – важнейшее свойство многих криптографических алгоритмов.

Обратимся сейчас к *коллизиям*. Эти, понятные с логической и с семантической точек зрения, термин и явление являются следствием того, что множество возможных сообщений всегда будет превышать множество возможных хеш-функций, поскольку длина последних ограничена (чаще всего эта длина составляет от 128 до 512 битов). Это означает, что коллизии неизбежны, по крайней мере, с теоретической точки зрения. Принято коллизии разделять на два типа.

Определение 10.4. *Коллизией 1-го рода* считаем ситуацию, при которой для данного сообщения M и для иного произвольного сообщения M' ($M \neq M'$) имеем $h(M) = h(M')$, вычисленные с использование одной и той же хеш-функции (или алгоритма хеширования).

Приведем для лучшего понимания простую аналогию. Из закрытой емкости с разноцветными шарами мы выбрали шар определенного цвета, например красного (M). Далее наугад достаем другой шар (M'). Если он будет окрашен в тот же цвет, что и первый (в красный), то мы будет считать такое совпадение коллизией 1-го рода.

Определение 10.5. *Коллизией 2-го рода* считаем ситуацию, при которой для двух произвольных сообщений M и M' ($M \neq M'$) имеем $h(M) = h(M')$, вычисленные с использование одной и той же хеш-функции (или алгоритма хеширования).

Следуя вышеописанной аналогии, в данном случае *одновременно* извлекаются два шара. Коллизия наступает при их одинаковой окраске.

Рассматриваемая схема ЭЦП состоит из трех основных шагов:

- генерация ключа;
- формирование подписи; для заданного электронного документа M (с помощью закрытого ключа) вычисляется подпись;
- проверка (верификация) подписи; для данных документа и подписи с помощью открытого ключа определяется действительность подписи.

10.3. Однонаправленные хеш-функции и алгоритмы хеширования

10.3.1. Алгоритм хеширования MD4

Для понимания сущности хеширования целесообразно проанализировать реальный алгоритм, например MD4.

Алгоритм был спроектирован Р. Ривестом (в то время профессором Массачусетского технологического института) в 1990 году. Описание алгоритма и пример реализации можно найти в документе **RFC 1320**[24]. Это один из алгоритмов целого семейства MD (message digest). К этому семейству принадлежат также алгоритмы MD2 и MD5. Но MD2 довольно старый алгоритм и сейчас практически не используется, а MD5 – это следующий шаг после MD4. В его основе лежит все тот же MD4.

Алгоритм построения цифрового дайджеста сообщения MD4 прост в реализации и обеспечивает построение «отпечатка» для сообщения произвольной длины. Предполагается, что сложность построения двух сообщений с одинаковым дайджестом имеет порядок 2^{64} операций, и что сложность построения сообщения по заданному цифровому дайджесту имеет порядок 2^{128} операций.

Результатом хеширования произвольного (входного) сообщения будет 128-битная последовательность. Весь алгоритм условно можно разделить на 5 стадий:

- расширение входного сообщения;
- разбивка расширенного сообщения на блоки;
- инициализация начальных констант (константы *A*, *B*, *C*, *D* или MD-буфер);
- обработка сообщения поблочно (основная процедура алгоритма хеширования);
- вывод результата.

Рассмотрим кратко каждую стадию, опираясь на указанный документ (RFC 1320).

Стадия 1. Расширение входного сообщения. Сообщение расширяется таким образом, чтобы его длина (в битах) была конгруэнтна 448 по модулю 512, т. е. сообщение расширяется до размера

[24] Rivest R. The MD4 Message-Digest Algorithm, RFC 1320 // IETF tools [Электронный ресурс]. URL: https://tools.ietf.org/html/rfc1320.

так, что ему недостает всего 64 бита, чтобы иметь длину, кратную 512. Расширение сообщения производится всегда, даже если его длина уже конгруэнтна 448 по модулю 512. Это значит, что если к размеру сообщения в битах после расширения добавить 64 (512 – 448), то полученная величина должна быть кратна 512, т. е. должна делиться на 512 без остатка.

Пример. Пусть дано сообщение «мир». Его длина (l) в битах равна 24 ($3 \cdot 8 = 24$). Расширив сообщение до 448 бит, мы и получим выполнение требуемого условия: 448 + 64 = 512; 512 / 512 = 1, остаток равен 0.

Расширение сообщения выполняется следующим образом: единственный единичный бит («1») добавляется в конец сообщения, а затем сообщение дополняется таким количеством «0» бит, чтобы его длина стала конгруэнтна 448 по модулю 512. В целом, к сообщению будет добавлено от одного до 512 битов.

Продолжая рассматривать пример сообщения «мир», расширим его, добавив единичный бит и 423 (448 – 24 – 1 = 423) нулевых бита.

Возникает естественный вопрос: что же делать с оставшимися 64 битами? Они предназначены для дальнейшего расширения сообщения, теперь уже до размера, кратного размеру блока, т. е. 512 битов. В этих 64 битах следует указать двоичное представление длины исходного сообщения («мир»). Еще раз отметим, что эта длина составляет 24 бита ($3 \cdot 8$). 64-разрядное представление величины l (т. е. длины сообщения до того, как оно было расширено) добавляется в конец результата, полученного на предыдущем шаге. Причем вначале дописывается младшее 32-разрядное слово, а затем старшее. В том случае, если величина l больше, чем 2^{64}, добавляются только младшие 64 бита величины l.

Для рассматриваемого нами примера значащими (младшими) двоичными символами десятичного числа 24 будут 11000. Эта комбинация должна быть впереди дополнена до 64-битного комплекта нулями.

Таким образом, *мы получили 512-битное расширение входного сообщения,* которое имело длину 24 бита.

Понятно, что при больших начальных размерах входного сообщения количество 512-битных блоков будет отличным от 1.

Стадия 2. Разбивка расширенного сообщения на блоки. Итак, мы знаем, что расширенная длина сообщения кратна 512. Это означает, что в таком формате дальнейшей обработке подвер-

гается сообщение длиной $n \cdot 512$ битов, где n – целое число, равное или большее 1.

На этой стадии каждый 512-битный блок разделяется на 16 32-разрядных слов ($16 \cdot 32 = 512$). Именно 32-разрядное слово является основной информационной и структурной единицей рассматриваемого алгоритма.

Стадия 3. Инициализация констант. Для вычисления цифрового дайджеста используются 4 переменных (буфер размером в четыре слова), обозначаемых символами *A*, *B*, *C*, *D*. Таким образом, здесь каждое из *A*, *B*, *C*, *D* может отождествляться с *32-разрядным регистром*. Эти регистры инициализируются следующими *шестнадцатеричными* величинами или начальными константами (младший байт указан первым):

A: 01 23 45 67
B: 89 ab cd ef
C: fe dc ba 98
D: 76 54 32 10.

Приведенные константы также называются *переменными сцепления*.

Стадия 4. Обработка сообщений 512-битными блоками. Указанные выше регистры *A*, *B*, *C*, *D* содержат текущее состояние хеша (текущий дайджест) обрабатываемого блока, а в конце обработки – итоговый хеш сообщения. Каждый регистр содержит 32-разрядную беззнаковую величину, поэтому *размер хеша будет равен 128* ($32 \cdot 4 = 128$) *битам*.

Каждый блок данных M_j преобразуется в 3 раундах. Схематично это преобразование представлено на рис. 10.1 и 10.2.

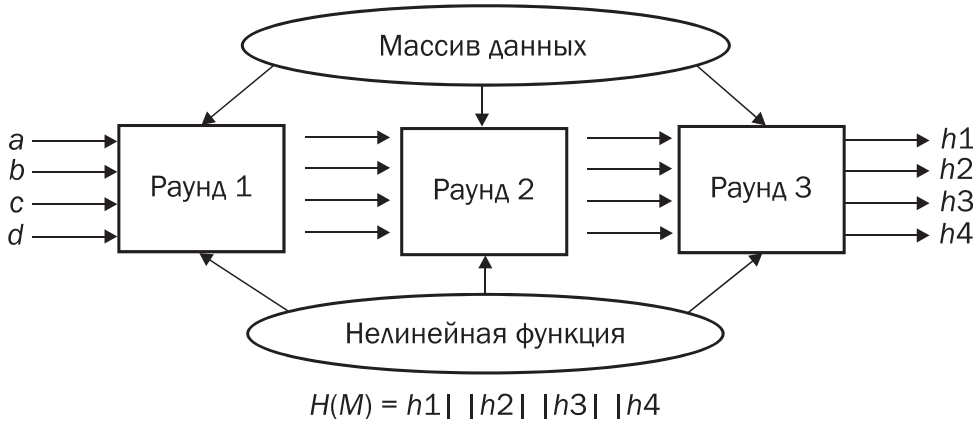


Рис. 10.1. Общая схема хеширования MD4

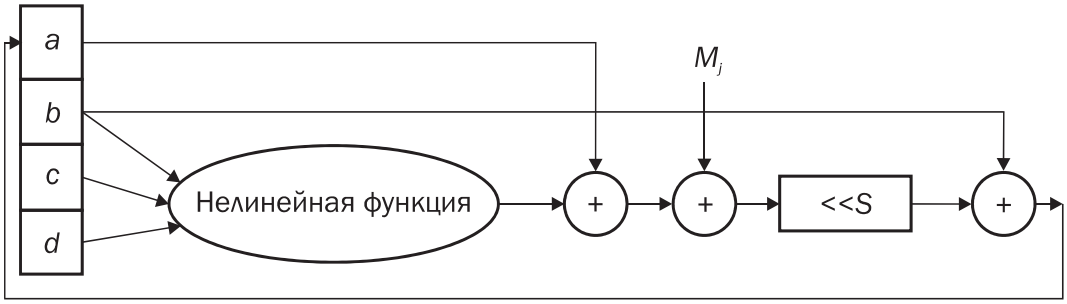


Рис. 10.2. Схема хеширования блока данных M_j

В каждом из указанных раундов используется отдельная функция преобразования: в первом раунде – $F(x, y, x)$, во втором – $G(x, y, z)$, в третьем – $H(x, y, z)$:

$$\left.\begin{array}{c} F(x, y, x) = (x\ AND\ y)\ OR\ (NOT\ x\ AND\ z), \\ G(x, y, z) = (x\ AND\ y)\ OR\ (x\ AND\ z)\ OR\ (y\ AND\ z), \\ H(x, y, z) = x\ XOR\ y\ XOR\ z. \end{array}\right\} \quad (10.1)$$

Вспомним, что $x\ XOR\ y$ означает сложение по модулю два битов x и y; $NOT\ x$ означает инверсию текущего значения x.

Таблицы истинности приведенных логических функций выглядят следующим образом (рис. 10.3)

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x	y	z	H
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Рис. 10.3. Таблицы истоинности используемых функций

Каждая из функций получает на вход три 32-разрядные величины и формирует один 32-разрядный результат.

Процессор (назовем так блок или модуль программы, выполняющий анализируемую обработку) также выполняет три базовые операции преобразования. Эти операции используют определенные выше функции F, G и H и выражаются следующими формулами (с учетом (10.1)):

$$\left.\begin{aligned} a &= (a + F(b, c, d) + X[k]) <<< s, \\ a &= (a + G(b, c, d) + X[k] + 5A827999) <<< s, \\ a &= (a + H(b, c, d) + X[k] + 6ED9EBA1) <<< s, \end{aligned}\right\} \quad (10.2)$$

где a, b, c, d обозначают регистры A, B, C, D; F, G и H – рассмотренные выше функции (10.1); знак <<< – операцию циклического сдвига влево; s – количество разрядов сдвига; $X[k]$ – 32-разрядное слово, $k = 0, 1, 2, \ldots, 15$; во втором и третьем выражениях указаны 32-разрядные константы в шестнадцатеричной форме.

Последние выражения означают следующее. Например,

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

в применении к конкретным регистрам имеет следующий вид:

$$A = (A + F(B, C, D) + X[0]) <<< 3,$$

т. е. процессор – на основании содержимого регистров A, B, C и D, а также слова «0» (первое по счету из 16: от 0 до 15) 32-разрядное слово обрабатываемого 512-битного блока – строит новое (текущее) значение регистра A.

В применении к регистру D операция преобразования может выглядеть следующим образом:

$$D = (D + F(A, B, C) + X[1]) <<< 7.$$

Рассмотрим для примера особенности рассматриваемых операций в 1-м раунде. Пусть [*abcd k s*] обозначает базовую операцию

$$a = (a + F(b, c, d) + X[k]) <<< s.$$

Тогда данный раунд представляет собой, в соответствии с алгоритмом, следующую последовательность элементарных преобразований для каждого из 16 слов:

[*ABCD* 0 3] [*DABC* 1 7] [*CDAB* 2 11] [*BCDA* 3 19],
[*ABCD* 4 3] [*DABC* 5 7] [*CDAB* 6 11] [*BCDA* 7 19],
[*ABCD* 8 3] [*DABC* 9 7] [*CDAB* 10 11] [*BCDA* 11 19],
[*ABCD* 12 3] [*DABC* 13 7] [*CDAB* 14 11] [*BCDA* 15 19].

Стадия 5. Вывод результата. 128-разрядным хешем входного сообщения (цифровым дайджестом) будет результат вывода содержимого регистров A, B, C, D, точнее говоря, конкатенация четырех 32-разрядных слов, начиная с младшего байта регистра A и заканчивая старшим байтом регистра D.

Рассмотренный алгоритм используется сейчас сравнительно редко в силу низкой криптостойкости. Более надежным является алгоритм MD5.

10.3.2. Особенности алгоритма MD5

Общая схема алгоритма преобразования блока сообщения приведена на рис. 10.4.

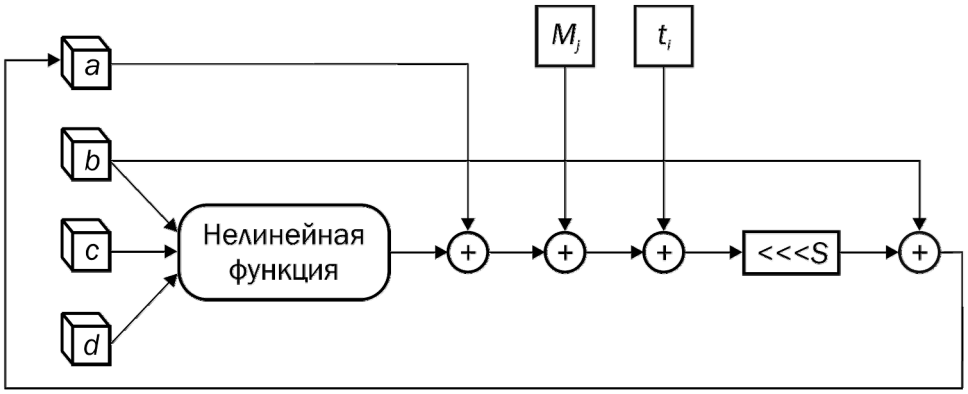


Рис. 10.4. Общая схема алгоритма преобразования блока данных на основе MD5

Основные отличия между алгоритмами MD4 и MD5 заключаются в следующем.

1. MD5 имеет на один раунд больше – 4 против 3 у MD4.
2. Чтобы уменьшить влияние входного текста, была введена уникальная константа для каждого раунда: $t[i]$.
3. Во втором раунде функция $G(x, y, z)$:

$$(x\ AND\ y)\ OR\ (x\ AND\ z)\ OR\ (y\ AND\ z)$$

заменена на

$$(x\ AND\ z)\ OR\ ((y\ AND\ (NOT\ z))$$

для того, чтобы сделать $G(x, y, z)$ менее симметричной.

4. На каждом шаге используется значение, полученное на предыдущем шаге. Это дает более быстрое изменение результата при изменении входных данных.
5. Для ускорения *лавинного эффекта* значения циклического сдвига в каждом раунде оптимизированы. Для этих же целей количество сдвигов различается от раунда к раунду и выбрано так, чтобы еще более увеличить этот эффект.

6. Изменен порядок, в котором обрабатываются слова в раундах 2 и 3 для того, чтобы сделать их менее похожими друг на друга.

10.3.3. Алгоритмы хеширования семейства SHA

Другим известным семейством алгоритмов хеширования является SHA (Secure Hash Algorithm – защищенный (безопасный) алгоритм хеширования). Как и MD5, SHA являются, по сути, улучшенными продолжениями MD4 (в наибольшей степени это относится к SHA-1). Алгоритм изначально разрабатывался национальным институтом стандартов и технологий США для его использования совместно со стандартом ЭЦП DSS. Как и в алгоритмах MD, хеширование на основе SHA предусматривает выполнение предварительных преобразований.

В таблице приведены основные характеристики алгоритмов класса SHA.

Характеристики алгоритмов хеширования SHA

Вариации алгоритма		Размер выходного хеша, бит	Промежуточный размер хеша, бит	Размер блока, бит	Максимальная длина входного сообщения, бит	Размер слова, бит	Количество раундов	Найденные коллизии
SHA-0		160	160	512	$2^{64} - 1$	32	80	Есть
SHA-1		160	160	512	$2^{64} - 1$	32	80	252 операции
SHA-2	SHA-256/224	256/224	256	512	$2^{64} - 1$	32	64	Нет
	SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80	Нет

Обратимся, например, к алгоритму SHA-1. Как видим из таблицы, он выдает 160-битное число. Для этого используются и инициируются изначально пять 32-разрядных констант:

A: 67 45 23 01,
B: fef c dab 89,
C: 98 ba dc fe,
D: 10 32 54 76,
E: c3 d2 e1 f0.

Вначале эти константы копируются в соответствующие переменные: A – в a, B – в b и т. д.

Главный цикл состоит из четырех раундов, каждый из которых включает по 20 операций (вспомним – в алгоритмах MD таких операций 16). Каждая такая операция предусматривает вычисление нелинейной функции над тремя переменными из набора a, b, c, d, e. После этого производятся операции сдвига и сложения, аналогичные вышерассмотренным, с использованием константы t. В рассматриваемом алгоритме применяется следующий набор функций:

$$\left.\begin{array}{c} F(x, y, x) = (x\ AND\ y)\ OR\ (NOT\ x\ AND\ z) \text{ для } t \text{ от 0 до 19,} \\ G(x, y, z) = x\ XOR\ y\ XOR\ z \text{ для } t \text{ от 20 до 39,} \\ H(x, y, z) = (x\ AND\ y)\ OR\ (x\ AND\ z)\ OR\ (y\ AND\ z) \\ \text{для } t \text{ от 40 до 59,} \\ I(x, y, z) = x\ XOR\ y\ XOR\ z \text{ для } t \text{ от 60 до 79.} \end{array}\right\} \quad (10.3)$$

Как видим, во всех рассматриваемых алгоритмах используются похожие функции.

Характеризуя криптостойкость рассмотренных алгоритмов, Б. Шнайер (Прикладная криптография, М.: Триумф, 2003. 816 с.) делает следующий вывод.

Алгоритм SHA, по сути, совпадает с алгоритмом MD4, отличаясь наличием расширяющего преобразования, дополнительным циклом обработки и улучшенным лавинным эффектом. Алгоритм MD5 – это улучшенный MD4.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Дайте определение хеш-функции.
2. Что такое «однонаправленность» хеш-функций и какова роль этого свойства хеш-функций в криптографии?
3. Что такое «коллизия»?
4. Дайте общую характеристику алгоритмам хеширования семейства MD.
5. Из каких основных стадий состоит алгоритм хеширования сообщения?
6. Рассчитайте общую длину хешируемого сообщения после предварительной стадии на основе алгоритма MD, если объем ис-

ходного сообщения составлял 0; 484; 512; 1000; 2000; 16 000 битов. Какова в каждом случае будет длина хешированного сообщения?

7. Представьте и охарактеризуйте структурную схему одного раунда алгоритмов хеширования на основе MD4; MD5; SHA-1.

Глава 11
ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ НА ОСНОВЕ ХЕША СООБЩЕНИЯ

11.1. ЭЦП при использовании RSA

В подглаве 9.1 мы упоминали о возможности реализации основных функций ЭЦП с помощью простого шифрования сообщения на основе RSA. Подтверждение подлинности и целостности полученного сообщения, которое зашифровано ключом отправителя, основывается исключительно на принадлежности ключа шифрования отправителю сообщения (тайного ключа отправителя: чисел d и n). Это означает следующее: если получатель зашифрованного сообщения после его расшифрования публичным ключом отправителя получает логически стройный документ, то это является гарантом целостности документа и указанием на авторство. Как видим, в данном случае ЭЦП, как самостоятельный и дополнительный элемент к сообщению, не генерируется (аналогия с использованием посредника).

Один из классических алгоритмов генерации ЭЦП на основе RSA отличается от рассмотренного случая лишь тем, что отправитель шифрует не само сообщение M (см. пункт 5.6.2), а его хеш $h(M)$. Понятно, что предварительно должна быть выполнена процедура генерации ключа отправителем.

Генерация ЭЦП или зашифрование $h(M)$:

$$s = (h(M))^d \bmod n, \qquad (11.1)$$

здесь пара чисел (d и n) – тайный ключ отправителя.

После этого число s, являющееся ЭЦП, присоединяется к сообщению M и пересылается получателю.

Получатель имеет в «своем распоряжении»: сообщение M, подпись к нему s, известный алгоритм хеширования. *Верификация подписи* предусматривает выполнение последовательных операций:

- *восстановление* $h(M)$:

$$s^e \bmod n = h'(M); \qquad (11.2)$$

- *вычисление хеша* для полученного сообщения: $h''(M)$ (для строгости анализа применяем новое обозначение);
- *сравнение хешей*: $h'(M)$ и $h''(M)$; только при равенстве сравниваемых величин сообщение с подписью будет однозначно аутентифицировано.

Авторство сообщения может быть установлено и доказано по паре ключей (d, n; e, n) с использованием *процедуры сертификации* (более подробно об этой процедуре речь пойдет ниже). Злоумышленник не сможет подменить сообщение (точнее, ему будет очень трудно это сделать), поскольку для этого необходимо вместо сообщения M подставить другое сообщение M', имеющее такое же значение хеша (хеш-функции), что и M, что является, как мы это подчеркивали выше, вычислительно трудной задачей. По этой же причине злоумышленник не сможет применить перехваченную подпись s для подписи другого документа, поскольку для другого документа будет получено иное значение хеша.

Таким образом, все необходимые свойства подписи описанным алгоритмом обеспечиваются, что же касается криптостойкости метода ЭЦП, то она определяется криптостойкостью используемого асимметричного криптографического метода и функции однонаправленного шифрования. Необходимо отметить также, что само сообщение M передается в открытом виде. Для того чтобы обеспечить конфиденциальность передаваемой в нем информации, требуется использование дополнительного шифрования по одной из известных симметричных или асимметричных схем.

Одним из аспектов использования RSA для ЭЦП является свойство, называемое *восстановлением сообщения* (подобно восстановлению хеша, см. формулу (11.2)). В этом случае зашифрованию тайным ключом отправителя подлежит не хеш сообщения M, а само сообщение M:

$$s = M^d \bmod n. \qquad (11.3)$$

Получателю направляется пара: M и s; как и выше, s – это ЭЦП.

Процедура *восстановления сообщения* получателем:

$$s^e \bmod n = M'. \qquad (11.4)$$

Подчеркнем отдельно важную деталь: **в алгоритме зашифрования/расшифрования сообщения на основе алгоритма RSA и в алгоритме генерации/верификации ЭЦП на основе того же алгоритма используются различные ключи: в первом случае – ключи получателя (открытый – при зашифровании и закрытый – при расшифровании), во втором – ключи отправителя (закрытый – при зашифровании, т. е. при генерации ЭЦП, и открытый – при расшифровании, т. е. для верификации)**.

Такой же механизм использования ключевой информации свойственен и другим подобным алгоритмам.

11.2. Схемы ЭЦП на основе проблемы дискретных логарифмов

Практически все известные алгоритмы ЭЦП на основе хеша сообщения являются разновидностями общей схемы цифровой подписи, использующей известную нам *проблему дискретного логарифма*. К означенному семейству относятся, в том числе, ЭЦП *DSA*, *Эль-Гамаля* и *Шнорра*, которые мы и рассмотрим ниже.

Формально все указанные разновидности подписи описываются следующими рассуждениями.

Выберем большие простые числа p и q, при этом q должно быть равным либо $p - 1$, либо простому множителю $p - 1$. Затем выбираем число g в диапазоне от 1 до p, для которого выполняется условие:

$$g^q = 1 \bmod p.$$

Все эти числа открыты и могут использоваться группой лиц.

Открытым ключом является число y:

$$y = g^x \bmod q, \qquad (11.5)$$

тайным – x (для сравнения рекомендуем обратиться к пункту 5.6.3).

Для создания ЭЦП сообщения M выбирается случайное число $k < q$; причем числа k и q – взаимно простые.

Далее вычисляется число r:

$$r = g^k \bmod p. \tag{11.6}$$

Обобщенное уравнение подписи имеет вид:

$$a \cdot k = (b + c \cdot x) \bmod q. \tag{11.7}$$

И *обобщенное уравнение проверки подписи*:

$$r^a = (g^b \cdot y^c) \bmod p. \tag{11.8}$$

11.2.1. Алгоритм цифровой подписи DSA

В августе 1991 года американским институтом стандартизации ((NIST) был утвержден стандарт DSS (Digital Signature Standard – стандарт цифровой подписи) для использования в алгоритме ЭЦП DSA (Digital Signature Algorithm – алгоритм цифровой подписи).

В алгоритме используются следующие параметры: p – простое число длиной от 64 до 1024 битов (число должно быть кратно 64); q – 160-битный простой множитель $p - 1$.

Далее вычисляется число g:

$$g = v^{(p-1)/q} \bmod p,$$

где v – любое число, меньшее $p - 1$, для которого выполняется условие

$$v^{(p-1)/q} \bmod p > 1.$$

Числа p, q, v могут использоваться группой лиц.

Открытый ключ y вычисляется в соответствии с выражением

$$y = g^x \bmod p, \tag{11.9}$$

где $x < q$; x – *закрытый ключ*.

Понятно, что открытый ключ размещается в общедоступной базе данных.

Вспомним, что в алгоритме подписывается не само сообщение M, а его хеш: $h(M)$. Для хеширования сообщений используется алгоритм SHA.

После генерации ключей их обладатель может подписывать свои сообщения.

Генерация ЭЦП: выбирается из условия, описанного выше, число k. Подпись состоит из пары чисел, r и s:

$$r = (g^k \bmod p) \bmod q, \tag{11.10}$$

$$s = (k^{-1} (h(M) + x \cdot r)) \bmod q. \tag{11.11}$$

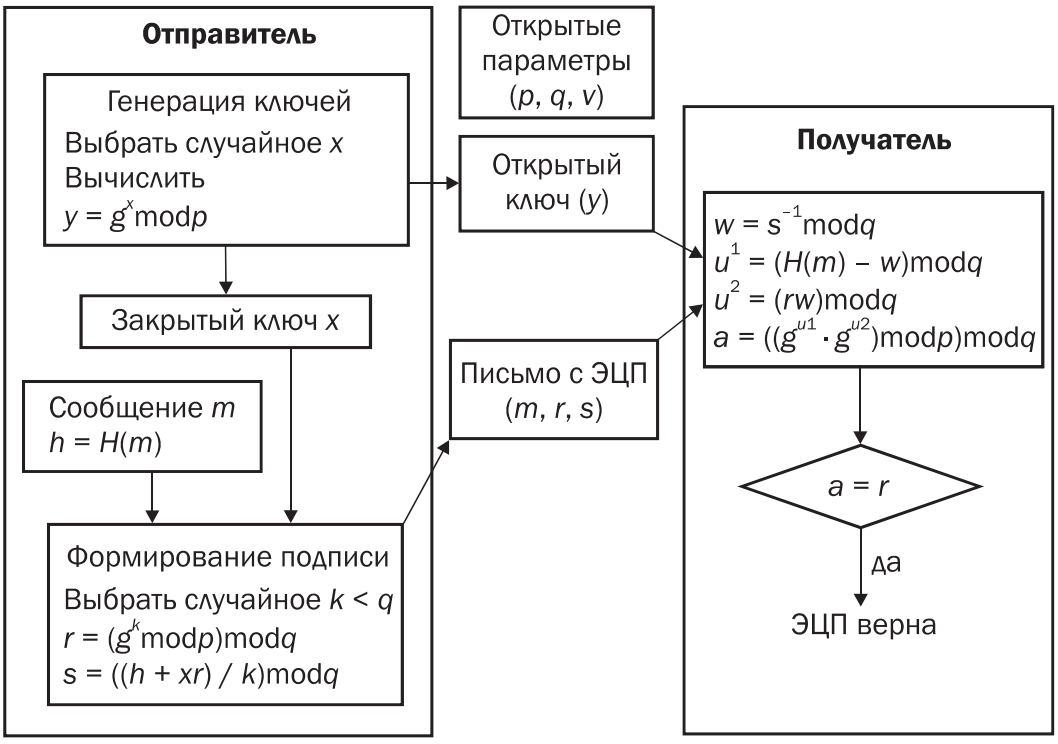
Как видим, соотношения (11.10), (11.11) незначительно отличаются от (11.6) и (11.7).

Для проверки подписи получатель подписанного сообщения (M, r, s) выполняет вычисления:

$$\left.\begin{aligned} w &= s^{-1} \bmod q, \\ u_1 &= (h(M) \cdot w) \bmod q, \\ u_2 &= (r \cdot w) \bmod q, \\ a &= ((g^{u_1} \cdot y^{u_2}) \bmod p) \bmod q. \end{aligned}\right\} \tag{11.12}$$

Подпись считается достоверной, если $a = r$.

На рисунке приведена общая схема алгоритма.



Общая схема алгоритма ЭЦП DSA

Быстродействие практической реализации рассматриваемого алгоритма можно существенно увеличить с помощью предвари-

тельных вычислений. Обратим внимание на то, что r не зависит от подписываемого сообщения M. Можно создать массив чисел k и рассчитать r для каждого k. Кроме того, можно вычислить значение k^{-1} для каждого k. Далее брать очередную пару r и k^{-1} для очередного сообщения M. По данным Б. Шнайера[25], такие предварительные вычисления по времени сопоставимы с процедурой верификации подписи. При этом длительность операций вычисления s составляет лишь несколько процентов от длительности всей процедуры генерации ЭЦП.

11.2.2. Схема подписи Эль-Гамаля

Схему Эль-Гамаля можно использовать как для шифрования (см. пункт 5.4.3), так и для цифровых подписей. В сравнении, например, с ЭЦП на основе RSA рассматриваемая схема обеспечивает более высокое быстродействие.

Вспомним, что *ключ* состоит из четырех чисел. Для его генерации нужно выполнить следующие действия.

1. Выбирается простое число p и два случайных числа, меньших, чем p: числа x и g.

2. Далее вычисляется

$$y = g^x \bmod p.$$

Открытый ключ: y, g и p; тайный ключ: x.

Чтобы подписать сообщение M, обладатель используемых для ЭЦП ключей должен выбрать, как и в предыдущей схеме, случайное число k, взаимно простое с $p - 1$. Затем вычисляются числа a и b, являющиеся цифровой подписью:

$$a = g^k \bmod p; \tag{11.13}$$

для вычисления b с помощью расширенного алгоритма Евклида (см. листинг на с. 55–56) решается уравнение

$$M = (x \cdot a + k \cdot b) \bmod (p - 1). \tag{11.14}$$

Для верификации подписи нужно убедиться, что выполняется равенство

$$y^a \cdot a^b = g^M \bmod p. \tag{11.15}$$

[25] Шнайер Б. Прикладная криптография. М.: Триумф, 2003. 816 с.

Как видим, в *классическом алгоритме ЭЦП по схеме Эль-Гамаля используется не хеш сообщения M, а подписываемое сообщение целиком*, которое следует представлять также числом.

Пример. Положим, что $p = 11$, $g = 2$, $x = 8$, вычисляем третье значение открытого ключа:

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

Таким образом, открытый ключ: $p = 11$, $g = 2$, $y = 3$; закрытый ключ: $x = 8$.

Подпись сообщения M ($M = 5$). Выбирается случайное число $k = 9$, взаимно простое с $p - 1 = 10$; вычисляется в соответствии с выражением (11.13) первый элемент ЭЦП:

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

Второй элемент ЭЦП находим на основе уравнения (11.14):

$$M = (x \cdot a + k \cdot b) \bmod (p - 1) = 5 = (8 \cdot 6 + 9 \cdot b) \bmod 10.$$

Решением будет число $b = 3$.

Подписанное сообщение: $M = 5$, $a = 6$, $b = 3$.

Проверка подписи в соответствии с уравнением (11.15):

$$(3^6 \cdot 6^3) \bmod 11 = 2^5 \bmod 11 = 10.$$

В ЭЦП DSS и Эль-Гамаля для каждой новой подписи необходимо использовать новое значение k. Если третья сторона добудет два сообщения, подписанные с использованием одного и того же k, ее шансы на взлом закрытого ключа отправителя значительно возрастают. Это обстоятельство мы подчеркивали при рассмотрении схем шифрования по Эль-Гамалю.

11.2.3. ЭЦП на основе схемы Шнорра

На основе схемы К. Шнорра (Claus Schnorr) строятся протокол аутентификация и ЭЦП. При этом подписывается, как и в алгоритме DSA, не сообщение M, а его хеш-функция на основе M.

В алгоритме используются следующие открытые параметры:

p – простое число в диапазоне от 512 до 1024 битов;

q –160-битное простое число, делитель $p - 1$;

любое число z ($z \neq 1$) такое, что

$$z^q = 1 \bmod p.$$

Числа p, z, q являются открытыми и могут применяться группой пользователей.

Выбирается число $x < q$ (x является тайным ключом) и вычисляется открытый ключ:

$$V = z^{-x} \bmod p. \tag{11.16}$$

Секретный ключ имеет длину не менее 160 битов.

Для *подписи сообщения M* выбирается случайное число k ($k < q$) и вычисляется число a:

$$a = z^k \bmod p. \tag{11.17}$$

Далее вычисляется хеш от канкатенации сообщения M и числа a:

$$H = h(M||a)$$

и вычисляется значение

$$y = (k + x \cdot H) \bmod q. \tag{11.18}$$

Получателю отправляются M, H, y; понятно, что ЭЦП составляют два последних элемента этого сообщения.

Проверка подписи: получатель вычисляет

$$a' = (z^y \cdot v^H) \bmod p; \tag{11.19}$$

затем он проверяет, равно ли хеш-значение H от конкатенации M и x':

$$H' = h'(M||a')$$

Подпись достоверна, если $H = H'$.

Существующий стандарт ЭЦП в Республике Беларусь (СТБ 34.101.45-2013) основан на схеме Шнорра (а также на эллиптических кривых)[26].

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. В чем состоит основное отличие криптосистемы передачи/получения зашифрованного сообщения от передачи/получения подписанного сообщения?

2. На чем основана криптостойкость ЭЦП Эль-Гамаля, Шнорра и DSA?

---

[26] Российские стандарты ЭЦП (ГОСТ Р 34.10–2001 и актуальный – ГОСТ Р 34.10–2012) также основаны на эллиптических кривых.

3. Представьте схемы ЭЦП Эль-Гамаля, Шнорра, взяв за основу схему на рисунке на с. 166.

4. Взяв ключевую информацию из примера 5.4, подпишите сообщение, состоящее из собственного имени, алгоритмом RSA.

5. Взяв ключевую информацию из примера 5.6, подпишите сообщение, состоящее из собственного имени, алгоритмом Эль-Гамаля.

Глава 12

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Мы достаточно подробно (см. пункт 5.4.4) проанализировали особенности практического применения в криптографии эллиптических кривых, задаваемых над полями Галуа. Для ясности вспомним важнейшие особенности.

Пусть задано простое число $p > 3$. Тогда эллиптической кривой E, определенной над простым конечным полем F_p, называется множество пар чисел (x, y) (точка с координатами (x, y) принадлежит F_p), которые удовлетворяют тождеству:

$$y^2 = (x^3 + a \cdot x + b) \bmod p, \qquad (12.1)$$

где $a, b \in F_p$ и $(4 \cdot a^3 + 27 \cdot b^2) \neq 0 \bmod p$.

Кроме того, к эллиптической кривой добавляется бесконечно удаленная точка O. Таким образом, точки, удовлетворяющие уравнению кривой E, и точка O образуют конечную абелеву группу.

Для точек эллиптической кривой определена операция сложения. Для двух точек, принадлежащих кривой E, $P(x_p, y_p)$ и $Q(x_q, y_q)$, точка $R(x_r, y_r)$, являющаяся их суммой, также будет лежать на эллиптической кривой (см. рис. 5.4 на с. 83).

Графически удвоение точки можно получить, построив касательную к точке и отразив точку пересечения касательной с эллиптической кривой относительно оси OX. Отсюда очевидно, что можно определить операцию умножения некоторой точки эллиптической кривой на целое число, которая позволяет определить

точку $Q = k \cdot P$ (точка P, умноженная на целое число k, обращается в точку Q).

Скалярное умножение осуществляется посредством нескольких комбинаций сложения и удвоения точек эллиптической кривой.

Пример 12.1. Некоторая точка $25 \cdot P$ может быть представлена как

$$25 \cdot P = (2 \cdot (2 \cdot (2 \cdot (2 \cdot P))) + 2 \cdot (2 \cdot (2 \cdot P))) + P.$$

Именно с **операцией умножения точки на целое число напрямую связана идея, надежность и криптостойкость эллиптической криптографии**.

Мы уже особо отмечали, что *задача вычисления дискретного логарифма на эллиптической кривой заключается в отыскании целого числа x по известным точкам P и* $Q = x \cdot P$ и является трудноразрешимой.

Помимо уравнения, важным параметром кривой является *базисная (генерирующая) точка G*, выбираемая для каждой кривой отдельно.

Секретным ключом в соответствии с технологией эллиптических кривых является большое случайное число *x*, а сообщаемым открытым ключом – *произведение x на базисную (генерирующую) точку G*.

Выбор базисной точки обусловлен тем соображением, чтобы ее порядок (для примера см. табл. 5.10 на с. 89) был достаточно большим: $2^{254} < n < 2^{256}$. Вспомним: **точка $P \in E$ называется точкой порядка n, если $n \cdot P = O$**.

Для того чтобы лучше понять практику использования рассматриваемой технологии, проанализируем весь алгоритм, взяв за основу существующий стандарт ЭЦП в России.[27]

Пример 12.2. Будем использовать следующие обозначения:

$H(M)_l$ – множество всех двоичных векторов длиной l битов – множество хеш-функций подписываемых сообщений; $l = 256$ битов (либо 512 битов);

V^* – множество всех двоичных векторов произвольной конечной длины;

p – простое число, $p > 3$;

---

[27] ГОСТ Р 34.10–2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. Введ. 01.01.2013. М.: Стандартинформ, 2013. 24 с.

M – сообщение пользователя, $M \in V^*$;

a, b – коэффициенты эллиптической кривой;

P – точка эллиптической кривой порядка q; в рассматриваемом стандарте q выбирается из условия: $2^{254} < q < 2^{256}$; $q \cdot P = O$ (точка P соответствует генерирующей точке кривой и является открытым параметром));

d – целое число – тайный ключ подписи; $0 < d < q$;

Q – точка эллиптической кривой – ключ проверки подписи:

$$Q = d \cdot P; \tag{12.2}$$

ζ – цифровая подпись под сообщением M.

Используемая эллиптическая кривая задается соотношением (12.1). Координаты точек при $x_1 \neq x_2$ вычисляются по формулам (5.17), а при выборе кривой следует руководствоваться рекомендациями, которые изложены в подпункте 5.4.4.2 настоящего пособия.

Как видим, каждый пользователь должен иметь тайный криптографический ключ (число d) и ключ проверки или верификации подписи, определяемый точкой эллиптической кривой Q с координатами (x_q и y_q), удовлетворяющей выражению (12.2).

Формирование (генерации) ЭЦП. Состоит из 6 операций.

1. Вычислить хеш сообщения M: $h(M)$; $h(M) \in H(M)_l$.

2. Вычислить целое число t, двоичным представлением которого является $h(M)$, и далее вычислить

$$e = t \bmod q. \tag{12.3}$$

Если $e = 0$, то принять $e = 1$.

3. Выбрать случайное (псевдослучайное) число k; $0 < k < q$.

4. Вычислить точку (координаты точки) C кривой E:

$$C = k \cdot P, \tag{12.4}$$

определить число r:

$$r = x_c \bmod q, \tag{12.5}$$

где x_c – x-координата точки C. Если $r = 0$, *вернуться к шагу 3*.

5. Вычислить значение s:

$$s = (r \cdot d + k \cdot e) \bmod q, \tag{12.6}$$

при $s = 0$ следует повторить шаг 3.

6. Определить двоичное представление чисел r и s, конкатенация которых ($r||s$) и будет является цифровой подписью ζ под сообщением M.

Верификация подписи.

1. Выполнить операцию, обратную по отношению к шагу 6 генерации ЭЦП: восстановление чисел r и s.

2. Вычислить хеш полученного сообщения M: $h'(M)$.

3. Повторить шаг 2 режима генерации ЭЦП: вычислить целое число t, двоичным представлением которого является $h'(M)$ (формально это значение может приводить к различным результатам здесь и при использовании формулы (12.3)) и далее вычислить

$$e' = t \bmod q. \tag{12.7}$$

Если $e' = 0$, то принять $e' = 1$.

4. Вычислить

$$v = (e')^{-1} \bmod q. \tag{12.8}$$

5. Определить значения двух чисел:

$$z_1 = (s \cdot v) \bmod n, \qquad z_2 = (-r \cdot v) \bmod q. \tag{12.9}$$

Здесь полезно обратиться к формуле (5.14).

6. Вычислить точку C эллиптической кривой:

$$C = z_1 \cdot P + z_2 \cdot Q \tag{12.10}$$

и определить значение числа r':

$$r' = x_c \bmod q. \tag{12.11}$$

7. Подпись верна, если $r' = r$.

Таким образом:

• исходными данными алгоритма генерации ЭЦП являются: сообщение *M*, алгоритм и параметры вычисления хеш-функции сообщения *M*, значение тайного ключа отправителя *d*, тип эллиптической кривой, случайное число *k*, с помощью которого определяется точка *C* кривой, точки эллиптической кривой: точка *P* порядка *q* и точка *Q*, удовлетворяющая выражению (12.2);

• исходными данными алгоритма верификации ЭЦП являются: полученное сообщение *M*, электронная подпись ζ сообщения, точка *P* порядка *n* и точка *Q*.

В анализируемом стандарте ЭЦП рассмотрен пример с конкретными числовыми значениями, который проанализируем здесь с некоторыми сокращениями.

Параметру p присвоено следующее значение: $p > 0{,}5 \cdot 10^{77}$ или точно

p = 57896044618658097711785492504343953926634992332820282019728792003956564821041;

коэффициенты эллиптической кривой: $a = 7$ и $b > 0{,}4 \cdot 10^{77}$ или точно

b = 4330887654676727690576590459565093199594211179445103958325296884203384958041 4;

порядок группы точек эллиптической кривой: число, большее $0{,}5 \cdot 10^{77}$ или точно

57896044618658097711785492504343953927082934583725450622380973592137631069619;

координаты точки P: $x_p = 2$, $y_p > 4 \cdot 10^{76}$ или точно

y_p = 40189740565390375033354494229370597756357393899055450806909793652134315662 80;

q = 57896044618658097711785492504343953927082934583725450622380973592137631069619;

тайный ключ отправителя сообщения: $d > 0{,}5 \cdot 10^{77}$ или точно

d = 55441196065363246126355624130324183196576709222340016572108097750006097525544;

ключ Q проверки подписи – координаты точки имеют следующие значения: $x_q > 0{,}5 \cdot 10^{77}$ или точно

x_q = 57520216126176808443631405023338071176630104906313632182896741342206604859403;

$y_q > 1{,}7 \cdot 10^{77}$ или точно

y_q = 17614944419213781543809391949654080031942662045363639260709847859438286763994.

Генерация ЭЦП. Пусть после выполнения шагов 1–2 в соответствии с формулой (12.3) было получено следующее числовое значения e:

e = 20798893674476452017134061561508270130637142515379653289952617252661468872421;

на шаге 3 выбрано число k:

k = 53854137677348463731403841147996619241504003434302020712960838528893196233395;

при этом точка $C = k \cdot P$ имеет координаты:

x_c = 29700980915817952874371204983938256990422752107994319651632687982059210933395;

y_c = 32842535278684663477094665322517084506804721032454543268132854556539274060910;

параметр $r = x_c \bmod q$ (формула (12.5)) принимает значение:

r = 29700980915817952874371204983938256990422752107994319651632687982059210933395;

параметр $s = (r \cdot d + k \cdot e) \bmod q$ соответственно равен:

s = 57497340027008465417892531001914703845522704264909856393371899917551583955 2.

Цифровой подписью (ζ) будет бинарная последовательность, состоящая из конкатенированных величин ($r||s$), представленных в соответствующей системе счисления.

Верификация подписи. На входе получатель подписанного сообщения имеет M и ζ как основные компоненты для верификации.

После выполнения третьего шага получим e' (формула (12.7)):

$e' = 207988936744764520171340615615082701306371425153796532899526172526614688724 21$;

как видим в процессе передачи подписанного документа его целостность не нарушена ($e' = e$).

Параметр $v = (e')^{-1} \bmod q$ (выражение (12.8)) принимает значение:

$v = 176866836059344686773017138249002685627468830806754967152880365724311457189 78$;

на очередном шаге алгоритма вычисляем:

$z_1 = 376991675009019385568410572935126561088413451904919426193045324127437209997 59$;

и

$z_2 = 141719984273434721125159179695007657692466558389728621144999326533336710922 1$;

точка $C = z_1 \cdot P + z_2 \cdot Q$ (формула (12.10)) будет иметь координаты:

$x_c = 297009809158179528743712049839382569904227521079943196516326879820592109333 95$;

$y_c = 328425352786846634770946653225170845068047210324545432681328545565392740609 10$;

наконец, вычисляем $r' = x_c \bmod q$:

$r' = 297009809158179528743712049839382569904227521079943196516326879820592109333 95$.

Поскольку $r' = r$, подпись верифицирована.

Главным достоинством ЭЦП на основе эллиптических кривых является то, что они обеспечивают криптостойкость на существенно меньших по длине ключах (в сравнении со схемами, рассмотренными в главе 11), что положительно отражается на времени генерации и верификации ЭЦП. Например, ЭЦП на основе эллиптических кривых с длиной ключа 160 битов имеют одинаковую стойкость с системами DSA и Эль-Гамаля с длиной ключа 1024 бита.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Чем определяется криптостойкость ЭЦП на основе эллиптических кривых?

2. Какую ключевую информацию используют отправитель и получатель подписанного сообщения на основе эллиптической кривой?

3. Используя тайный ключ (d) из таблицы задания после пункта 5.6.3, подпишите сообщение, состоящее из собственного имени алгоритмом на основе ЭК, параметры которой определяются в примере 5.10.

4. В чем, по Вашему мнению, состоят преимущества схемы ЭЦП на ЭК в сравнении с традиционными? В чем заключаются их недостатки?

Глава 13
АТАКИ НА ЭЛЕКТРОННУЮ ЦИФРОВУЮ ПОДПИСЬ И ПРОТИВОДЕЙСТВИЕ АТАКАМ

13.1. Характеристика основных типов атак

Целью атак на цифровые подписи является, в конечном итоге, возможность их подделки или фальсификации. Известны модели атак на ЭЦП, описанные достаточно давно Ш. Голвассером, С. Мисали и Р. Ривестом[28], однако они актуальны и в настоящее время. Здесь мы найдем много схожего с атаками на шифры (см. подглаву 3.5). Кратко охарактеризуем эти атаки.

Атака с использованием открытого ключа. Имеется в виду ключ субъекта, подписывающего документ. *Криптоаналитик* обладает только открытым ключом.

Атака на основе известных сообщений. В распоряжении криптоаналитика имеются некоторые ЭЦП и соответствующие им документы.

Адаптивная атака на основе выбранных сообщений. Криптоаналитик может получить подписи электронных документов, которые он выбирает сам.

Основные результаты или цели описанных атак можно классифицировать следующим образом.

---

[28] Goldwasser Sh., Micali S., Rivest R. A digital signature scheme secure against adaptive chosen-message attacks // SIAM Journal on Computing. 1988. No. 17 (2). P. 281–308.

Полный взлом ЭЦП. Получение закрытого ключа отправителя подписанного сообщения. Это означает полный взлом алгоритма.

Универсальная подделка подписи. Криптоаналитик находит алгоритм, позволяющий подделывать подписи для любого электронного документа.

Выборочная подделка подписи. Дает возможность подделывать подписи для документов, выбранных криптоаналитиком.

Экзистенциальная подделка подписи. Дает возможность получения допустимой подписи для некоторого документа, не выбираемого криптоаналитиком.

Самой опасной атакой является адаптивная атака на основе выбранных сообщений. При анализе алгоритмов ЭЦП на криптостойкость принято принимать во внимание и детально анализировать именно ее.

Современные алгоритмы генерации и использования ЭЦП оставляют криптоаналитику мало шансов на получение закрытого ключа алгоритма из-за вычислительной сложности задач, на основе которых ЭЦП построена. Более вероятен поиск *коллизий*, сущность которых мы рассмотрели при анализе хеш-функций. С учетом применения хеш-функций во многих основных схемах ЭЦП нахождение коллизий для алгоритма подписи эквивалентно нахождению коллизий для самих хеш-функций. Таким образом, в контексте обсуждаемой проблемы мы возвращаемся к коллизиям первого и второго рода.

Коллизия первого рода эквивалентна экзистенциальной подделке, а коллизия второго рода – выборочной.

Подделка документа (коллизия первого рода). Криптоаналитик подбирает документ M' к данной подписи. Однако в подавляющем большинстве случаев такой документ может быть только один. Это оригинальный документ M, для которого и создавалась данная ЭЦП. Причина здесь кроется в следующем. Во-первых, документ представляет из себя осмысленный, логически стройный текст. Во-вторых, текст документа оформлен по установленной форме (как правило, каждый тип электронного документа в каждой стране должен создаваться на основе установленных правил, формируемых на базе национального законодательства; например, у нас в стране базовым документом является *Закон Республики Беларусь «Об электронном документе и электронной цифровой подписи» от 28 декабря 2009 г. № 113-З*). Это значит, если для фаль-

шивого набора байт (M') и произойдет коллизия с хешем исходного документа (M), то должны выполниться 3 следующих условия:

• случайный документ должен соответствовать сложно структурированному формату документа;

• случайный документ должен представлять собой текст, оформленный по установленной форме;

• текст документа M' должен быть осмысленным и грамотным (по определению).

Вероятность такой коллизии ничтожно мала.

Получение двух документов с одинаковой подписью (коллизия второго рода). Вероятность такого события является более высокой в сравнении с вероятностью наступления коллизии первого рода. В этом случае злоумышленник «фабрикует» два документа (M и M') с одинаковой ЭЦП и в нужный момент подменяет один документ другим. При использовании криптостойкой хэш-функции такая атака должна быть также вычислительно сложной. Однако эти угрозы могут реализоваться из-за слабостей конкретных алгоритмов хэширования, подписи или ошибок в их реализациях.

В заключение анализа атак на цифровые подписи укажем на существование так называемых «социальных» атак. Социальные атаки направлены не на взлом алгоритмов ЭЦП, а на манипуляции с открытым и закрытым ключами отправителей подписанных сообщений. Эффективным средством противодействия таким атакам являются *сертификационные центры*, которые выполняют и иные функции.

13.2. Сертификаты и сертификационные центры

Использование надежных протоколов обмена ключами и защита закрытого ключа от несанкционированного доступа позволяет снизить опасность упомянутых выше социальных атак.

Так как открытый ключ доступен любому пользователю, то необходим механизм гарантии того, что этот ключ принадлежит именно указанному владельцу. Необходимо обеспечить доступ любого пользователя к подлинному открытому ключу любого другого легитимного пользователя, защитить эти ключи от подмены злоумышленником, а также организовать отзыв ключа в случае его компрометации.

Проблемы защиты ключей от подмены решается с помощью *сертификатов* или *сертификационных центров (СЦ)*, функции которых, в первом приближении, похожи на функции посредника, которые мы рассмотрели при анализе соответствующего алгоритма ЭЦП (см. подглаву 9.2).

Структура сертификатов (или *сертификатов открытых ключей*) является важной частью известного **протокола X.509**. В соответствии с эти документом основными элементами сертификата являются:

- данные о владельце и его открытый ключ;
- период действия сертификата;
- выдавшая сертификат организация.

Центр сертификации формирует закрытый ключ и собственный сертификат, формирует сертификаты конечных пользователей и удостоверяет их аутентичность своей цифровой подписью. Также центр проводит отзыв истекших и компрометированных сертификатов и ведет базы выданных и отозванных сертификатов. Обратившись в СЦ, можно получить собственный сертификат открытого ключа, сертификат другого пользователя и узнать, какие ключи отозваны.

В Республике Беларусь действуют несколько лицензированных СЦ (у нас они чаще называются удостоверяющими (УЦ)). К их числу относятся, например, СЦ Республиканского унитарного предприятия «Информационно-издательский центр по налогам и сборам» (http://pki.by/), Удостоверяющий центр таможенных органов Республики Беларусь (http://gtk.gov.by/ru/eldeclaration_new/udost_centr), Удостоверяющий центр министерства промышленности Республики Беларусь (http://cniitu.by/cert), Удостоверяющий центр национального центра маркетинга и конъюнктуры цен (http://ca.ncmps.by/articles/view/8), Удостоверяющий центр Белорусской товарной биржи (http://ecp.by/?page=280) и др.

Для получения сертификата между Центром и клиентом заключается договор (соглашение). Формально сертификат оформляется в виде соответствующего документа (карточка открытого ключа). На рис. 13.1 приведен фрагмент одного из таких документов.

Как мы ранее упоминали, значение открытого ключа хранится в защищенной базе данных, а программное обеспечение для генерации ЭЦП с тайным ключом передается клиенту на носителе.

КАРТОЧКА ОТКРЫТОГО КЛЮЧА

Наименование организации владельца открытого ключа: *Республиканское унитарное предприятие "Информационно-издательский центр по налогам и сборам"*
Страна: *BY*
Населенный пункт: *Минск*
Адрес: *пр-т Машерова, 7, к.123*
Общие данные: *Корневой удостоверяющий центр РУП "Информационно-издательский центр по налогам и сборам"*
Адрес электронной почты: *support8ca.info-center.by*

Использование ключа:
Подписание CRL, Подписание сертификата, Согласование ключа, Шифрование данных, Шифрование ключа, Неотрекаемый, Цифровая подпись

Дополнительные атрибуты ключа:
Идентификатор открытого ключа (2.5.29.14): 6883 5885 005D 6CB8 5924 27D9 8DD2 DE4C 9F71 9E28
Бланк карточки открытого ключа (1.3.6.1.4.1.12656.8): 1.3.6.1.4.1.12656.8.5.2

Срок действия открытого ключа:
Начало: 13.03.2009 08:42:50 (GTM+2) Окончание 13.03. 2024 23: 59:59 (GTM+2)
Алгоритм: *СТБ 1176.2-99 / РД РБ DH*
Значение открытого ключа:
30820124 30818F06 092B0601 0401E270 01230381 810292C4 BFF70B03 846D620E E5652003 3C6F6C63
3806001F B6A08B43 812E2190 EDCB4975 4CF32AAD 7899F820 1F1F9496 A84F6362 14036C33 38D7C696
C7FCD8DF 5B06E8CA 532623A6 248E0481 3BAF193D 06021791 40CFF434 58F41BE6 F973AE57 184C5762
1DF0ACAD 2733286F A8B8161A 135222D3 2DD3B899 B91BF67C 3E18EFDB D8383081 8F06092B 06010401
E2700120 03818102 169B1B16 658F2C47 F3EDA754 CBA2B464 1BA774A4 238983F4 607A1DAC 2428548A
781E0BBC A0D29503 A59698B0 FAA4EF34 C98E7D41 9BC5513E 7A24521C EBA14B0A C535EC4B 59CB8D71
7AE918C4 EB7AC573 08E06E87 A632DDFB 662E1021 4976607A 90366EE0 5D20FC7C 82CAC194 B4AEAFBC
0F88386A B0DD1EEA 38AD5370 C1C609C4

Параметры алгоритма:
идентификатор объекта согласно РД КБ РБ 07040.1206-20041.3.6.1.4.1.12656.7.2

Подпись владельца открытого ключа: *Начальник отдела*

Рис. 13.1. Пример оформления карточки открытого ключа

Закрытый ключ является наиболее уязвимым компонентом всей криптосистемы цифровой подписи. Злоумышленник, укравший закрытый ключ пользователя, может создать действительную цифровую подпись любого электронного документа от лица этого пользователя. Поэтому особое внимание нужно уделять способу хранения закрытого ключа.

В настоящее время используются следующие устройства хранения закрытого ключа: смарт-карты, USB-брелоки, таблетки Touch-Memory. Кража или потеря такого носителя может быть легко замечена пользователем, после чего соответствующий сертификат должен быть немедленно отозван. Из перечисленных устройств, вероятно, наибольшую степень защиты обеспечивает смарт-карта, поскольку ее использование требует ввода PIN-кода, т. е. имеет место двухфакторная аутентификация. После успешно-

го окончания процедуры аутентификации подписываемый документ или его хэш передается в карту. Встроенный процессор осуществляет процедуры генерации ЭЦП. В процессе формирования подписи не происходит копирования закрытого ключа, поэтому все время существует только единственная его копия – на носителе.

13.3. SSL-сертификаты

Как известно, одним из сегментов информационного пространства, где проблема безопасного обмена сообщениями со взаимной идентификацией участников процесса стоит особенно остро, является взаимодействие клиента с *web-узлами* или *web-приложениями*. В наибольшей степени это касается электронной торговли (или *Интернет-магазинов*), налоговой сферы и др.

В контексте рассматриваемой проблемы кратко охарактеризуем *бизнес-модели* данной сферы деятельности. Основными из таких моделей считаются:

бизнес – бизнес (business-to-business, B2B); под этим обычно понимается любая деятельность одних компаний по обеспечению других компаний сопроводительными услугами, дополнительным оборудованием, а также товарами, предназначенными для производства других товаров; примером деятельности на основе этой модели может быть работа *аутсорсинговых* фирм-резидентов Парка высоких технологий (ПВТ) в Республике Беларусь;

бизнес – потребитель (business-to-consumer или business-to-client, B2C); наиболее характерным примером является как раз *Интернет-торговля*;

потребитель – потребитель (consumer-to-consumer, C2C); *Интернет-аукционы*, *Интернет-площадки*;

бизнес – администрация (business-to-administration, B2A); предполагает взаимодействие, например, между коммерческими и государственными структурами (вероятно, на основе этой модели осуществляется взаимодействие между администрацией ПВТ и фирмами-резидентами ПВТ);

потребитель – администрация (consumer-to-administration, C2A); примером может служить взаимодействие государственных структур и потребителей в социальной и налоговой сфере (например, *электронное декларирование*).

Для того чтобы в каждом из рассмотренных случаев данные в момент передачи из браузера на сервер невозможно было перехватить, используется специальный протокол *https*, который шифрует все передаваемые данные. А для того чтобы активировать возможность работы протокола *https*, как раз и нужны цифровые *SSL-сертификаты* (SSL – Secure Socket Layer, уровень защищенных сокетов).

SSL-сертификаты – самый распространенный на данный момент тип сертификатов в Интернете. Из названия сертификата понятно, что он строится на основе **протокола SSL**. SSL изначально разработан компанией *Netscape Communications* для добавления протокола *https* в свой веб-браузер Netscape Navigator. Впоследствии на основании протокола SSL 3.0 был разработан и принят *стандарт RFC*, получивший имя *TLS* (Transport Layer Security – безопасность транспортного уровня).

На рис. 13.2 показано размещение **протокола SSL** в стеке протоколов TCP/IP. Его архитектура состоит из двух уровней протоколов:

• *протокол записи* SSL;

• три протокола более высокого уровня:

– *протокол квитирования* (Handshake Protocol);

– *протокол изменения параметров шифрования* (Change Cipher Spec Protocol);

– *протокол извещения* (Alert Protocol).

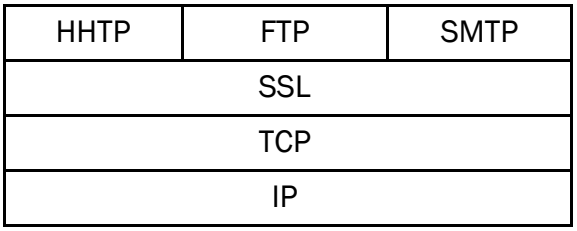


Рис. 13.2. Размещение SSL в стек протокола TCP/IP

Архитектура и стек протоколов SSL представлены на рис. 13.3.

Протокол квитирования отвечает за организацию *сеанса* (session) между клиентом и сервером. На его основе осуществляется взаимная аутентификация сторон, согласовываются алгоритмы шифрования и криптографические ключи (на основе известного *алгоритма рукопожатия*).

<table>
<tr><td>Протокол квитирования SSL</td><td>Протокол изменения параметров шифрования SSL</td><td>Протокол извещения SSL</td><td>HTTP</td></tr>
<tr><td colspan="4">Протокол записи SSL</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

Рис. 13.3. Архитектура и стек протоколов SSL

Протокол изменения параметров шифрования генерирует однобайтовое сообщение, которое дает указание начать копирование параметров состояния ожидания в текущее состояние, что приводит к обновлению комплекта шифров, используемых для данного соединения.

Протокол извещения предназначен для передачи другой участвующей в обмене данными стороне извещений, касающихся работы SSL.

Протокол записи (это уровневый протокол) обеспечивает поддержку двух сервисов для *соединения* (connection) между узлами системы: *конфиденциальность* и *целостность сообщений*. Существуют четыре протокола записи: *протокол рукопожатия* (handshake protocol), *протокол тревоги* (alert protocol), *протокол изменения шифра* (the change cipher spec protocol), *протокол приложения* (application data protocol).

Для организации защищенного SSL-соединения на web-сервере необходимо установить SSL-сертификат. SSL-сертификаты выдаются известными нам удостоверяющими или сертификационными центрами. Центров сертификации существует достаточно много, вот перечень самых популярных:

• Comodo – работает с 1998 года, штаб-квартира расположена в Нью-Джерси, США;

• Symantec – самый крупный игрок на рынке SSL-сертификатов, владеет тремя крупнейшими центрами сертификации: Thawte (основан в 1985 году), Verisgin и Geotrust; штаб-квартира расположена в Калифорнии, США;

• Trustwave – основан в 1995 году, штаб-квартира расположена в Чикаго, США.

В Республике Беларусь продажей SSL-сертификатов занимается ряд фирм. Например «Хостер» (https://hoster.by/service/services-

and-solutions/ssl-certificates/), «Экстмедиа» (http://extmedia.by/ssl. desc. html), «Дримхост» (http://www.dreamhost.by/ ssl.html) и др.

Как правило, в сертификате хранится следующая информация:

- полное (уникальное) имя владельца сертификата;
- доменное имя, страна, город владельца;
- открытый ключ владельца (клиента);
- дата выдачи SSL-сертификата;
- дата окончания действия (легитимности) сертификата;
- полное (уникальное) имя центра сертификации;
- цифровая подпись издателя.

Важным компонентом рассматриваемой сертификационной системы является *корневой сертификат SSL* – одна из частей схемы ***PKI*** (Public Key Infrastructure, инфраструктуры публичных, или открытых, ключей), которая идентифицирует корневой центр сертификации.

Инфраструктура открытых ключей (PKI) – технология аутентификации с помощью открытых ключей. Это комплексная система, которая связывает открытые ключи с личностью пользователя посредством удостоверяющего центра.

В основе PKI лежит использование криптографической системы с открытым ключом и несколько основных принципов:

- закрытый ключ известен только его владельцу;
- удостоверяющий центр создает сертификат открытого ключа, таким образом удостоверяя этот ключ;
- никто не доверяет друг другу, но все доверяют удостоверяющему центру;
- удостоверяющий центр подтверждает или опровергает принадлежность открытого ключа заданному лицу, которое владеет соответствующим закрытым ключом.

Таким образом, PKI представляет собой систему, основным компонентом которой является удостоверяющий центр и пользователи, взаимодействующие между собой посредством удостоверяющего центра.

PKI реализуется в *модели клиент – сервер*, т. е. проверка какой-либо информации, предоставляемой данной службой, может происходить только по инициативе клиента.

Выделяют 5 видов архитектур PKI:

- простая;
- иерархическая;

- сетевая;
- кросс-сертифицированная;
- архитектура мостового удостоверяющего центра (УЦ).

Простая архитектура PKI. Все пользователи доверяют одному УЦ, через который осуществляется взаимодействие между ними.

Иерархическая архитектура PKI – это наиболее часто встречающаяся архитектура. Функционирует «под руководством» *корневого* (головного) УЦ. Частный пример иерархической PKI – корпоративная PKI.

Сетевая архитектура PKI – частный случай иерархической архитектуры. Но здесь нет корневого центра. В этой архитектуре все УЦ доверяют рядом стоящим УЦ, а каждый пользователь доверяет только тому УЦ, который выдал SSL-сертификат.

Архитектура кросс-сертифицированной корпоративной PKI может рассматриваться как гибрид иерархической и сетевой архитектур. Характеризуется самой сложной цепочкой выдачи и анализа средств сертификации.

Архитектура мостового УЦ разрабатывалась для сглаживания недостатков предыдущей архитектуры. В данном случае все компании (владельцы сертификатов) «доверяют» одному, *мостовому УЦ*, который является практически их головным УЦ, но также выступает в роли посредника между другими УЦ (белорусские центры выступают от лица международных центров выдачи SSL-сертификатов).

Основное отличие между разными центрами сертификации состоит в цене выдаваемых сертификатов, а также в том, в каком количестве браузеров установлен их *корневой сертификат*. Ведь если в браузере нет корневого сертификата этого центра, то посетитель с таким браузером получит ошибку при входе на сайт с сертификатом такого центра. Что касается перечисленных выше центров сертификации, то их корневые сертификаты установлены практически во всех существующих браузерах.

Чтобы проверить, корневые сертификаты каких центров сертификации установлены в вашем браузере, достаточно в настройках браузера найти данную опцию[29].

В настоящее время сертификационные центры используют ключи длиной 2048 битов на основе RSA.

---

[29] Полезную информацию о практическом использовании SSL-сертификатов и о СЦ можно найти на странице http://habrahabr.ru/company/tuthost/blog/150433/.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Классифицируйте атаки на ЭЦП.
2. Охарактеризуйте коллизии, которые могут возникнуть при использовании ЭЦП.
3. Что такое карточка открытого ключа?
4. Охарактеризуйте основные бизнес-модели.
5. Для чего создаются и как применяются SSL-сертификаты?
6. Охарактеризуйте структуру протокола SSL.
7. Дать сравнительную характеристику архитектур PKI.
8. Воспользовавшись кнопкой «Office» (находится в верхнем левом углу соответствующего приложения), добавьте цифровую подпись к документу, например MS Word.

Раздел V

МЕТОДЫ ОБФУСКАЦИИ В ЗАЩИТЕ ИНФОРМАЦИИ

Глава 14
ОСНОВНЫЕ МЕТОДЫ ЗАЩИТЫ ПРОГРАММНОГО КОДА

14.1. Общая сравнительная характеристика методов защиты программного обеспечения

Появляющиеся в доступных источниках информации сведения об атаках на продукты лидеров производства программного обеспечения (ПО) – наглядное доказательство того, что на сегодняшний день еще нет действительно эффективной защиты ПО. У этого есть ряд причин.

1. *Сложность* – использование сторонних компонентов, петель обратной связи, многовариантность реализации, взаимодействие с другими приложениями, неконтролируемые каналы передачи данных – учесть все это очень сложно.

2. *Баги* – уникальный вид ошибок программирования, которые возникают при плохой реализации того или иного решения, но, в отличие от ошибок, не мешают выполнению основной логики программы, а лишь снижают эффективность ее выполнения, делая нестабильной и непредсказуемой. По статистике, на 1000 строк кода приходится около 50 багов. Существует множество тестов, призванных выявить баги. Но даже после применения наиболее эффективных из них (например, QA – Quality Assurance) в каждой 1000 строк остается около 5 багов, что, учитывая объемы современных программ, например операционной системы Windows XP, насчитывает около 40 млн строк кода.

3. *Удобство атак*. Можно проводить атаки на расстоянии, организовывать распределенные атаки, подбирать ключи с помощью

мощных вычислительных систем, использовать одни и те же приемы взлома для схожих видов ПО, применять известные виды атак, пользуясь тем, что большинство пользователей крайне медлительны в установке обновлений.

4. *«Бесплатность» воспроизводства*. Специфика хранения программных продуктов делает стоимость их копирования практически нулевой.

5. *Популярность*. В современном мире практически ни одна сфера жизнедеятельности не обходится без использования ПО.

Могут различаться *цели атак*: пиратство, разработка конкурентного программного продукта, порча либо даже получение контроля над работой ПО. Универсальной защиты не существует, поэтому выбор того или иного подхода при защите зависит от цели и объекта атаки. Более того, чаще всего наиболее надежные методы защиты не являются оптимальным выбором по ряду причин:

• дороговизна (покупка, внедрение, обслуживание); затраты на применение защитных мер должны соответствовать потерям при взломе с учетом их вероятности;

• ограничения возможностей использования (чем надежней защита, тем меньше функций может выполнять ПО);

• повышенные требования к соблюдению правил информационной безопасности при работе с ПО.

Существуют различные *критерии анализа программного продукта для успешного выбора средств защиты: цель вероятного взлома, уровень профессионализма потенциальных взломщиков, доступные для взлома ресурсы (программные, аппаратные, временные) и др.*

Одним из направлений защиты ПО является *защита кода*, поскольку его анализ считают одной из ключевых составляющих при взломе. Среди методов защиты можно выделить следующие:

• использование *цифровых водяных знаков* (ЦВЗ, software watermark), основывающееся на записи в код программы скрытой информации (кому принадлежат авторские права и т. д.), которая позволяет истинному автору программы доказать то, что она является именно его интеллектуальной собственностью (но обычно использование водяных знаков не ограничивается только этим);

как видим, эта технология во многом схожа с применением стеганографических методов;

• *отпечатки пальца* (software fingerprint) – технология, которая кроме записи информации, позволяющей доказать право собственности на программу, требует записи в каждую копию программы уникального идентификационного кода, присваиваемого каждому покупателю программы, что позволяет впоследствии отследить нарушителя авторского права;

• определение *подлинности кода* (tamper-proofing) – в код программы помещается процедура проверки целостности самой программы, что позволяет определить, была ли программа изменена (были ли внесены какие-то либо изменения в ее код); если эта процедура обнаруживает, что в программу внесены изменения, она делает программу нефункциональной; упомянутые процедуры могут строиться, например, на основе известных нам методов хеширования или кодов четности (CRC);

• *зашифровывание кода программы*, после чего она в зашифрованном виде поставляется конечным пользователям (иногда эффективно зашифровывать только наиболее важные, критические участки кода, а не весь код программы); когда пользователь запускает такую программу, вначале будет инициирована процедура ее расшифрования (с помощью некоторого ключа).

Недостатки перечисленных методов могут состоять в том, что у злоумышленника может появиться возможность подвергнуть изменению водяной знак или отпечаток пальца, либо после приобретения лицензионной копии программы произвести извлечение расшифрованных частей программы в процессе ее работы из памяти.

В большинстве случаев для обхода защиты взломщику требуется изучить принцип ее работы, то, как она взаимодействует с самой защищаемой программой. Этот процесс называется процессом *реверсивной (обратной) инженерии* или *реинжинирингом*.

В последнее время специалисты считают, что наиболее эффективным средством противодействия несанкционированному использованию или реинжинирингу ПО является *обфускация программного кода*. Именно этот класс методов и будет являться предметом нашего дальнейшего анализа.

14.2. Характеристика современных методов обфускации кода

Определение. *Обфускация* (от англ. *obfuscate* – делать непонятным, запутывать) – внесение в исходный код программы таких модификаций, которые не изменяют поведение программы, ее функциональность (заложенный алгоритм), но значительно затрудняют понимание указанного алгоритма.

Формальное определение задачи обфускации: придумать алгоритм, который, получая на вход некоторую программу P, преобразует ее в программу O(P) таким образом, что код программы O(P) не должен раскрывать информацию об алгоритмах, параметрах и внутренней структуре этой программы P.

В связи с этим в основу методов обфускации положены следующие принципы:

- добавление лишних операций, не влияющих на работу приложения;
- пустое ветвление;
- изменение названий объектов;
- удаление оформления, комментариев;
- использование «экзотических» возможностей языков разработки ПО;
- вставка запрещенных (недекомпилируемых) данных;
- разделение одной переменной на несколько или наоборот – хранение в одной переменной нескольких переменных;
- использование «темных» предикатов (opaque predicats);
- построение плоского потока управления и др.

Не любой код одинаково хорошо поддается обфускации. Например, нельзя вносить изменения в самовоспроизводящиеся коды (quine), часто нельзя изменить синтаксис стандартных функций или объектов с открытым (public) доступом. Синтаксис некоторых языков программирования чувствителен к пробельным знакам.

При разработке методов обфускации необходимо соблюдать требования:

- сохранять семантику (входные и выходные данные должны совпадать с данными до обфускации);
- повышать запутанность (код после обфускации должен быть сложнее для анализа и понимания);

• соблюдать баланс между эффективностью метода и ресурсными затратами на его применение.

14.2.1. Уровни обфускации

Программный код может находиться в нескольких формах: исходной, промежуточной, в машинном коде.

Обфускация на уровне исходного кода. Данный тип обфускации может использоваться, если необходимо защитить программу на этапе ее разработки (например, от недобросовестных офисных сотрудников); если программа распространяется в виде исходных кодов (клиентские скрипты языков программирования для Интернета, например JavaScript, VBScript и др.); если это серверные библиотеки или *CMS-системы*, предоставляемые заказчику с ограниченными возможностями использования или распространения.

На данном уровне исходный и защищенный код, P и $O(P)$, представлены одним языком программирования. Преимуществом создания алгоритмов запутывания на этом уровне является то, что нет необходимости изучать дополнительные сведения о трансляции или интерпретации кода. Все что нужно знать – это сам язык программирования.

Недостаток вытекает из преимущества: для изучения кода также не нужно изучать дополнительные тонкости языка. Кроме того, благодаря оптимизации современных компиляторов многие неиспользуемые участки кода просто не входят в сборку (в том числе и специальный «мусор», создаваемый некоторыми алгоритмами). Соответственно, специалисты, изучающие данный код на более низком уровне, не столкнутся со многими препятствиями. Дополнительные ограничения возникают из-за строгости семантики кода исходного уровня, не позволяющей приводить код к виду, содержащему нечитаемые символы в названиях объектов, использовать смежную область памяти и др.

Обфускация на промежуточном уровне. Особый вид программного кода появился благодаря технологии выполнения ПО на виртуальных машинах (NET, Java, Flash). Данный код является более сложным для понимания, чем исходный, так как переводит его команды в более универсальный набор. Тем не менее оперировать этими командами гораздо легче, чем дизассемблированным

кодом. Для таких команд созданы специальные декомпиляторы, позволяющие получить на их основе результат близкий или даже идентичный (в случае с Flash) исходному коду. Большинство новых программных продуктов разрабатывается именно для работы на виртуальных машинах. Проблема защиты кода данного уровня стоит наиболее остро.

Обфускация на машинном уровне. Является наиболее сложным механизмом обфускации. Для его реализации необходимо хорошо знать не только правила работы компилятора для данного языка, но и языки низкого уровня. Но именно на машинном уровне можно создать наиболее сложные для распутывания алгоритмы обфускации, которые могут не только затруднить изучение кода, но и вообще помешать получить данные для изучения с помощью каких бы то ни было инструментов реинжиниринга.

Особенности обфускации сетевых приложений. Применение обфускации к Интернет-приложениям имеет ряд особенностей, поскольку такие приложения обладают свойствами, не характерными для настольного программного обеспечения (ПО), а именно:

• массовый доступ – популярные Интернет-ресурсы в минуту обслуживают более 1000 пользователей, а при DDoS-атаках их количество измеряется миллионами;

• распределение частей приложения на клиентской и серверной стороне, а также неконтролируемый маршрут передачи информации между ними;

• распространение кода в открытом виде; это касается не только клиентской части приложения, но и серверной, которая выдается заказчику с ограниченными условиями пользования;

• специфика выполнения: кэш браузера, технология генерации на стороне сервера, ограничения протоколов и др.

Это обусловливает повышенные требования к производительности программного обеспечения, контролю его подлинности на этапе выполнения, проверке потока данных, учета специфики выполнения.

Большинство методов обфускации приводит к увеличению затрат ресурсов для выполнения программы, так как производит запутывание за счет внедрение дополнительного кода (добавляет лишнее ветвление, циклы, переменные и т. п.). Но есть и такие приемы, которые в результате не только затрудняют изучение

кода, но и делают его выполнение менее затратным. К их числу относятся:

• *минификация* – за счет удаления комментариев и необязательных пробельных элементов, а также за счет замены названий объектов на более короткие, но уже не несущие смысловой нагрузки, достигается существенное снижение размера программы, что до сих пор очень актуально при работе в сети;

• *многопоточность* – физически компоненты программы размещены на различных серверах и динамически подгружаются при обращении к ресурсу, что позволяет не только затруднить копирование, но и ускорить загрузку за счет распараллеливания трафика;

• *кэширование* – сохранение результатов обфускации в виде статических данных для последующего использования; актуально для динамических методов обфускации, которые выполняются каждый раз при обращении к программе; временное или частичное сохранение результатов их работы позволяет значительно снизить нагрузку на сервер, что является критичным для многих масштабных проектов.

14.2.2. Особенности применения известных методов обфускации кода

14.2.2.1. Запутывание потока выполнения

Проиллюстрируем на понятных примерах некоторые особенности данного метода.

Использование исключений. Идея заключается в том, чтобы использовать перехваченные исключения аналогично меткам перехода в операторе *goto*. Данная технология активно используется при обфускации Skype[30].

Пример 14.1. Например, простейшую проверку булевой переменной:

```
if(flag)
   code1
else
   code2
```

[30] CLEFIA // Википедия [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/CLEFIA.

можно преобразовать к следующему варианту:

```
try:
   x = 1 / (1 - flag)
   code2
except ZeroDivisionError:
   code1
```

Пример 14.2. Учитывая возможность фильтрации исключения по типу, аналогичным образом можно представить оператор *switch*, используя в качестве идентификаторов *case* различные типы исключений. Можно также использовать внешнюю обработку исключений:

```
function f2( ){
   trash-code
   x = 1 / (1 - flag)
   trash-code
}
function f1( ){
   try{
     f2( )
   }
   catch(){
     code1
   }
}
```

Это позволяет полностью вынести код логики из соответствующей функции, при этом сохранив ее необходимость.

Перемешивание. Идея заключается в разделении кода на блоки, расположенные в псевдослучайном порядке внутри оператора *switch*. Сам оператор *switch* находится внутри цикла, который срабатывает необходимое количество раз. Во время выполнения меняется переменная, определяющая ключ необходимого идентификатора в *case*.

Пример 14.3. Следующий код:

```
int x, y, z;
x=0;         // блок1
y=1;         // блок2
z=2;         // блок3
if(y<x) z=0;  // блок4
y=-1;        // блок5
if(y>x) z=1   // блок6
```

после перемешивания примет вид:

```
int x, y, z;
int step=6;
while(step)
{
 switch(step)
 {
   case 1: if(y<x) z=0;  step=2; break; // блок4
   case 2: y=-1;  step=5; break;        // блок5
   case 3: y=1;  step=4; break;         // блок2
   case 4: z=2;  step=1; break;         // блок3
   case 5: if(y>x) z=1; step=7; break;  // блок6
   case 6: x=0;  step=3;  break;        // блок1
   default: step = 0;
 }
}
```

Для усложнения анализа такого кода можно добавить «мусорные кейсы» (*case*) и пустые действия внутри блоков. Также можно реализовать менее очевидный механизм присвоения значений переменной *step*, в том числе и разновариантный (для чего необходим набор дублирующих друг друга *case*-операторов).

14.2.2.2. Непрозрачные предикаты

Анализируемый подход основан на *задаче выполнимости булевых формул* (SAT) из теории вычислительной сложности алгоритмических задач. Постановка задачи состоит в том, чтобы определить, можно ли назначить всем переменным, встречающимся в формуле, значения «ложь» или «истина» так, чтобы формула стала истинной.

Используемые положения:

• *проблема останова* – не существует общего алгоритма, позволяющего определить завершенность заданных действий при известных входных значениях по описанию алгоритма и входным данным;

• «*десятая проблема Гильберта*»[31] – неразрешимость диофантовых уравнений общего вида за конечное число операций при

[31] Состоит в нахождении универсального метода целочисленного решения произвольного алгебраического диофантова уравнения; см.: Матиясевич Ю. В. Десятая проблема Гильберта. М.: Наука, 1993. 128 с.

произвольных неизвестных и целых рациональных числовых коэффициентах; например, следующее диофантово уравнение не имеет решений в целых рациональных числах:

$$(x + 1) \cdot (x + 1) \cdot (x + 1) + (y + 1) \times$$
$$\times (y + 1) \cdot (y + 1) - (z + 1) \cdot (z + 1) \cdot (z + 1) = 0.$$

Пример 14.4. В данном примере никогда не выполнится код внутри *if*, поскольку это опровергнет известную теорему Ферма:

```
for (x in [1 .. random()]):
   for (y in [1 .. random()]):
      for (z in [1 .. random()]):
         for (n in [3 .. random()]):
            if (x ^ n + y ^ n == z ^ n):
               code
```

Пример 14.5. В этом примере, основанном на гипотезе Коллатца[32], наоборот, код внутри *if* будет заведомо выполнен:

```
x = random() + 2
while(x > 1):
   if(x % 2):
      x = x * 3 +1
   else:
      x = x / 2
   if(x == 1):
      code
```

Пример 14.6. Пример непрозрачных предикатов, автоматически генерируемых программой *Appfuscator*, выглядит так:

```
(x & 0x8e3ef800) != 0x70641deb && (uint)x / 0x9388ea != 0x3ab6921c
```

либо в более объемном представлении:

```
(x2 | x1 | x0) & (x2 | x1 | x4) & (x2 | x1 | x3) & (x2 | x0 | x4) & (x2 | x0
| x3) & (x2 | x4 | x3) & (x1 | x0 | x4) & (x1 | x0 | x3) & (x1 | x4 | x3) &
(x0 | x4 | x3) & (!x2 | !x4 | !x0) & (!x2 | !x4 | !x1) & (!x2 | !x4 | !x3) &
(!x2 | !x0 | !x1) & (!x2 | !x0 | !x3) & (!x2 | !x1 | !x3) & (!x4 | !x0 | !x1)
& (!x4 | !x0 | !x3) & (!x4 | !x1 | !x3) & (!x0 | !x1 | !x3)
```

[32] См. Стюарт И. Величайшие математические задачи. М.: Альпина нонфикшн, 2015. 460 с.

14.2.2.3. Субституция

Идея основана на обращении к полям и методам объекта как к ключам ассоциативного массива.

Пример 14.7. Строка кода

```
alert(document.getElementById("spec-id").innerHTML);
```

преобразовывается к виду:

```
a = "document";
b = "getElementById";
c = "innerHTML";
alert(this[a][b][c]);
```

Внести дополнительные затруднения в анализ, т. е. обфусцировать код, можно, например, кодируя или шифруя строковые значения (о сущности такого подхода мы кратко вспоминали в начале данной главы).

14.2.2.4. Кодирование строковых значений

Различное изменение «внешнего вида» кода, а также строковых значений позволяет затруднить использование прямого поиска для анализа кода. Рассмотрим некоторые особенности такого подхода.

Шестнадцатеричная кодировка. Проиллюстрируем ее примером.

Пример 14.8. Наиболее простым является кодирование строки в шестнадцатеричный вид. Например, защита от обнаружения вставки знака для защиты авторских прав, о которых мы говорили выше, может выглядеть так:

```
this["document"]["getElementById"]("spec-id")["innerHTML"] = "copyright";
```

После кодирования последний фрагмент кода может принять следующий вид:

```
this["x\64x\6fx\63x\75x\6dx\65x\6ex\74"]["x\67x\65x\74x\45x\6cx\65
x\6dx\65x\6ex\74x\42x\79x\49x\64"]("x\73x\70x\65x\63x\2dx\69x\64
")["x\69x\6ex\6ex\65x\72x\48x\54x\4dx\4c"] =
"x\63x\6fx\70x\79x\72x\69x\67x\68x\74";
```

Генерация данных. Более сложные преобразования можно осуществлять с помощью различных операций, приводящих к появлению строковых результатов. Например, чтобы сгенерировать строку «getElementById», можно использовать информацию об

исключении, вызванном при обращении к несуществующим функциям со специально подобранными именами.

Пример 14.9.

```
try{
   (getE(leme(ntB(yId()))))
} catch(e){
   x = (e+'').split('(').slice(1,5).join(''); // x = "getElementById"
}
```

Другой способ получения символов – манипулирование числовыми операциями. С учетом того что некоторые объекты могут неявно преобразовываться в числовые значения (пустой массив – 0, true – 1 и т. д.), можно получать произвольные числовые значения, не используя ни одного числового символа.

Пример 14.10. Чтобы получить число 123, можно применить следующую последовательность операций:

```
x = -~[]*(""+-~[]+-~-~[]+-~-~false-~~~true+~[]); //123
```

На таком приеме основана работа обфускатора-кодировщика *jsNoalnumCom*. Так, например, строка

```
alert(2*2);
```

будет закодирована в этом случае следующим образом:

```
([][(![]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[]+[][[]])[+!+[]+[+[]]]+(
![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!
+[]+[+[]]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]][([][(![]+[])[+[]]+(![]+[]+[][[]])[+!+[]
+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!
+[]]]+[])[!+[]+!+[]+!+[]]+(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[!+[]+!+[]
]]()[(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]
]+(!![]+[])[+[]]])(!+[]+!+[]*(+(!+[]+!+[])))
```

Другим вариантом преобразования является перевод символов в их кодовые значения, а затем представление этих значений в виде операций над переменными.

Пример 14.11. Приведенный выше код с выводом результата 2*2 будет преобразован обфускатором-кодировщиком *jjencode* к следующему виду:

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$
$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$
,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")
[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.
```

```
__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$
]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.__
_)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$.
__$+$.$$_+$._$_+$.__+"("+$._$_+"*"+$._$_+");"+"\"")())();
```

Аналогичным образом работает обфускатор-кодировщик *aaencode*, но использует при этом символы японских иероглифов, образующие смайл-знаки, как в следующем примере.

Пример 14.12.

```
ﾟωﾟﾉ= / ｀m´）ﾉ ~┻━┻   //*´∇｀*/ ['_']; o=(ﾟｰﾟ) =_=3; c=(ﾟΘﾟ)
=(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)= (o^_^o)/ (o^_^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟﾉ :
((ﾟωﾟﾉ==3) +'_') [ﾟΘﾟ] ,ﾟｰﾟﾉ :(ﾟωﾟﾉ+ '_')[o^_^o -(ﾟΘﾟ)]
,ﾟДﾟﾉ:((ﾟｰﾟ==3) +'_')[ﾟｰﾟ] }; (ﾟДﾟ) [ﾟΘﾟ] =((ﾟωﾟﾉ==3) +'_')
[c^_^o];(ﾟДﾟ) ['c'] = ((ﾟДﾟ)+'_') [ (ﾟｰﾟ)+(ﾟｰﾟ)-(ﾟΘﾟ) ];(ﾟДﾟ) ['o'] =
((ﾟДﾟ)+'_') [ﾟΘﾟ];(ﾟoﾟ)=(ﾟДﾟ) ['c']+(ﾟДﾟ) ['o']+(ﾟωﾟﾉ +'_')[ﾟΘﾟ]+
((ﾟωﾟﾉ==3) +'_') [ﾟｰﾟ] + ((ﾟДﾟ) +'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+ ((ﾟｰﾟ==3) +'_')
[ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [(ﾟｰﾟ) - (ﾟΘﾟ)]+(ﾟДﾟ) ['c']+((ﾟДﾟ)+'_')
[(ﾟｰﾟ)+(ﾟｰﾟ)]+ (ﾟДﾟ) ['o']+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ];(ﾟДﾟ) ['_'] =(o^_^o) [ﾟoﾟ]
[ﾟoﾟ];(ﾟεﾟ)=((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟДﾟ) .ﾟДﾟﾉ+((ﾟДﾟ)+'_') [(ﾟｰﾟ) +
(ﾟｰﾟ)]+((ﾟｰﾟ==3) +'_') [o^_^o -ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟωﾟﾉ +'_')
[ﾟΘﾟ]; (ﾟｰﾟ)+=(ﾟΘﾟ); (ﾟДﾟ)[ﾟεﾟ]='\\'; (ﾟДﾟ).ﾟΘﾟﾉ=(ﾟДﾟ+ ﾟｰﾟ)[o^_^o -
(ﾟΘﾟ)];(oﾟｰﾟo)=(ﾟωﾟﾉ +'_')[c^_^o];(ﾟДﾟ) [ﾟoﾟ]='"';(ﾟДﾟ) ['_'] ( (ﾟДﾟ)
['_'] (ﾟεﾟ+(ﾟДﾟ)[ﾟoﾟ]+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+
(ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+
((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ ((o^_^o) -
(ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ)
+ (ﾟΘﾟ))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+((o^_^o) +(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+
(ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((o^_^o)
+(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟΘﾟ)+
(ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (o^_^o))+ (o^_^o)+ (ﾟДﾟ)[ﾟoﾟ]) (ﾟΘﾟ)) ('_');
```

В обоих случаях используется неявный вызов функции выполнения кода из строки (*eval*) с передачей туда строки из кодировочных значений символов:

```
0["constructor"]["constructor"](
  "return alert(\"\\123\\143\\146\\162\\145\\164\")"
)(); // alert("secret");
```

Кодирование невидимыми символами. Предусматривает запись кодового значения символов с помощью невидимых знаков (табуляций, пробелов, переводов строки, возвратов каретки).

При выполнении кода происходит обратное преобразование к десятичной системе, после чего кодовые значения переводятся в символы и выполняются в *eval*.

Пример 14.13. Запись alert(2*2), реализованная в обфускаторе-кодировщике *js-invis* (для наглядности пробел заменен на 0, а табуляция – на 1) выглядит так:

```
1100001
1101100
1100101
1110010
1110100
101000
110010
101010
110010
101001
```

В последнем случае в качестве разделения используется перенос строки, благодаря чему можно не соблюдать одинаковую длину записи символов. Это выгодно при использовании большого количества символов, находящихся в начале кодировочной таблицы. Иначе будет экономней использовать равную длину, задействовав при этом возврат каретки и перенос строки как в качестве дополнительных символов для кодирования, получив таким образом систему счисления по основанию 4 и уменьшив тем самым на единицу порядок размерности длины.

14.2.3. Использование других форматов хранения

Усложнить анализ кода можно, разместив его фрагменты за пределами самого скрипта.

Атрибуты html-тегов. Рассмотрим пример.

Пример 14.14

```
<body><img src="./pict.jpg" a="al" b="er" c="t(1);"></body>
<script>
   a = document.body.innerHTML;
   eval(
       a.split('a="')[1].split('"')[0] +
       a.split('b="')[1].split('"')[0] +
       a.split(' c="')[1].split('"')[0]
   );
</script>
```

В результате выполнения данного скрипта в оператор *eval* будет передана строка *alert(1)*, полученная в результате представления содержимого *body* как строки с последующем ее разделением по атрибутам *a*, *b*, *c* и извлечением правых частей, из которых затем извлекается содержимое между кавычками аналогичным способом.

Упаковка в изображение. Технология хранения кода в виде изображения позволяет реализовывать его асинхронную загрузку, скрытие от мониторинга загружаемых javascript-файлов, а также сжатие кода.

Пример 14.15. Пример того, как выглядит реализованная в виде изображения библиотека *jquery*, представлен на рис. 14.1.

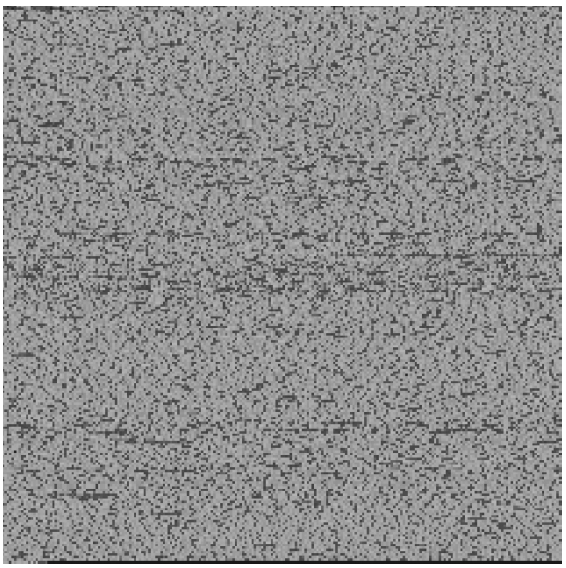


Рис. 14.1. Код библиотеки *jquery*, упакованный в изображение

Для восстановления кода пиксели изображения преобразуются в числовые значения символов:

```
var oData = ctxImg.getImageData(0,0,Width, Height).data;
var a = [];
var len = oData.length;
var p = -1;
for (var i=0;i<len;i+=4) {
  if (oData[i] > 0)
  a[++p] = String.fromCharCode(oData[i]);
};
var strData = a.join("");
```

14.3. АВТОРСКИЕ МЕТОДЫ ОБФУСКАЦИИ

14.3.1. Использование интерпретируемых языков программирования

Метод является серьезной модификацией известного *метода преобразования кода в последовательность операторов*. При этом новый метод свободен от основных недостатков, свойственных известному. Покажем это на примерах.

Пример 14.16. *Статическое расположение операторов*, выполняющих код, может выглядеть так:

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$
$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$
,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")
[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.
__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$
]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.__
_)[$.$_][$.$_];$.$($.$($.$$+"\""+!CODE!+"\"")())();
```

Данный подход к записи операторов не зависит от защищаемого кода и располагается в конструкции всегда одним и тем же образом. Вследствие этого может быть проведена атака, использующая шаблон замены данного участка на инструкцию вывода кода в поток консоли или файла.

Пример 14.17. *Разовое присвоение значений объектам* на протяжении всего выполнения кода, хранение всех значений в одном массиве выглядит следующим образом:

```
$ = {
   ___: ++$, // 0
   $$$$: (![] + "")[$], // f
   __$: ++$, // 1
   $_$_: (![] + "")[$], // a
   _$_: ++$, //2
   $_$$: ({} + "")[$], // b
   $$_$: ($[$] + "")[$], // d
   _$$: ++$, // 3
   $$$_: (!"" + "")[$], // e
   $__: ++$, // 4
   $_$: ++$, // 5
   $$__: ({} + "")[$], // c
```

```
    $$_: ++$, // 6
    $$$: ++$, // 7
    $___: ++$, // 8
    $__$: ++$ // 9
};
```

Подобный подход подвержен обратному декодированию, основанному на последовательном обходе ключей соответствующего массива.

Пример 14.18. *Преобразование символов в операторы по принципу «один к одному»*, простые приемы получения дополнительных значений за счет использования индекса строки и несложных для проверки инструкций позволяют провести деобфускацию путем комбинаций двух замен, основанных на регулярных выражениях:

```
regex = /"[^"]+?"\[\d+\]/g;
strCodes = str.match(regex);
for (var i in strCodes) {
try{
   str = str.replace(convert(strCodes[i]),"\""+eval(strCodes[i])+"\"");
}
catch(e) {}
}
regex = /\(([^()]+?)\)/g;
strCodes = str.match(regex);
for (var i in strCodes) {
try{
  str = str.replace(convert(strCodes[i]),"\""+eval(strCodes[i])+"\"");
}
catch(e) {}
}
```

Для устранения выявленных и продемонстрированных на примерах уязвимостей были использованы следующие идеи.

1. Разбиение исполняемого кода на несколько частей, перемешанных методом *ChenxiWang's*, как это показано в двух нижеследующих примерах.

Пример 14.19

```
intf(){
intx, y, z;
 x=0; //блок1
 y=1; // блок2
```

```
 z=2; // блок 3
 if(y<x) z=0; // блок4
 y=-1; // блок 5
 if(y>x) z=1 // блок 6
 return z;
```

Пример 14.20

```
intf(){
intx, y, z;
int step=6;
while(step)
{
 switch(step)
{
   case 1:   if(y<x) z=0;  step=2; break; // блок 4
case 2:   y=-1;  step=5; break;// блок 5
   case 3:   y=1;  step=4; break; // блок 2
   case 4:   z=2;  step=1; break; // блок 3
   case 5:   if(y>x) z=1; step=7; break; // блок 6
   case 6:   x=0;  step=3;  break; // блок 1
   default step = 0;
 }
 return z;}
```

2. Генерация строковых значений в отдельных функциях, что реализовано в следующем коде (показан пример функции для генерации строки по заданному набору чисел).

Пример 14.21

```
function genStr(ids){
string str="";
foreach(id in ids){
switch(id){
case 4: str+="0x6C"; break;
case 16: str+="0x65"; break;
case 133: str+="0x6F"; break;
…
case 8: str+="0x48"; break;
}
}
return str;
}
string str = genStr([8,16,4,4,133]); // Hello
```

3. Использование тождественных либо допустимых по точности математических операций для записи чисел.

Пример 14.22

```
y = DataTime.Now;
x = y%10+((y%100)/10)%10)-(y%100)%9+1;// x = 1;
```

4. Использование дополнительных приемов кодирования, основанных на субституции строк и генерации исключений.

Пример 14.23. Пример показывает кодирование методов с помощью *строковых оберток и исключений*:

```
document.getElementsByTagName("html")[0].innerHTML =
document.getElementsByTagName("body")[0].length;
a=document;
c='getElementsByTagName';
a[c]("html").innerHTML = a[c]("body")[0].innerHTML.length;
try{
(getE(leme(ntB(yId())))
}catch(e){
x = (e+'').split('(').slice(1,5).join(''); // x = "getElementById";
}
a = (alert+'').split("ive ")[1].substr(0,4); // a = "code";
b = this["\x65\x76\x61\x6C"]; // b = eval; //function
```

14.3.2. Метод шифрования данных

Поскольку поиск и изменение строковых данных в обфусцированом коде является одним из ключевых моментов атаки, в целях большей степени защиты можно использовать шифрование кода на ключе, задаваемом в виде *рекуррентного математического соотношения*.

Использование рекуррентных математических соотношений позволяет избежать коллизий с однозначностью перевода шифруемых символов, а также уязвимостей, обусловленных длиной ключа. Подобная уязвимость приводит к периодическим повторениям шифруемых участков, что особенно актуально при шифровании программного кода, обладающего большим объемом данных и часто повторяющимися последовательностями символов.

Итак, *сущность* предлагаемого подхода состоит в следующем. К некоторому однозначному числовому определению шифруемо-

го символа *A* (например, кодовому значению из используемой таблицы) добавляется некоторое число *F*(*X*), полученное в результате вычисления неизвестного (третьей стороне) математического выражения. В качестве переменных *X* при этом могут выступать известные данные, например порядковый номер символа (в шифруемом сообщении), кодовое значение предыдущего символа в этом сообщении.

Спецификой математической формулы является возможность получения одинаковых значений для ряда передаваемых переменных при различных наборах математических операций. Благодаря этому нельзя с уверенностью полагаться на то, что ключ найден, если с его помощью удалось расшифровать отдельный набор данных.

Но использование простых математических операций не представляет значительной сложности для операции криптоанализа (слома шифра). Поэтому важнейшей особенностью данного подхода являются динамические и рекуррентные преобразования. В качестве примера динамических преобразований можно привести операцию деления по модулю, изменяемому в процессе использования по заданному правилу, в качестве рекуррентного преобразования – зависимость одной части выражения от четности другой части выражения. Объединение динамического и рекуррентного подхода позволяет добиваться еще более сложных для обратного анализа действий, например деление по модулю, значение которого устанавливается в предыдущих частях выражения. Также добавлена возможность использования операции инкрементации.

Существует возможность составления формулы, специально подобранной для скрываемого текста, благодаря чему можно привести его к последовательности одинаковых символов.

Пример 14.24. Пусть дан текст «*абвгд*». Если его зашифровать с помощью формулы вида

```
x-2*(x==1)-4*(x==2)-6*(x==3)-8*(x==4),
```

то на выходе получим последовательность «*ааааа*».

Пример 14.25

Последовательность «*310a*», можно привести к виду «0000» с помощью формулы

```
-3*(x==0)-1*(x==1)-49*(x==3).
```

Поясним это.

Сдвиг для первого символа последовательности (3; нумерация формально начинается с нуля) составляет:

$$-3*(0==0)-1*(0==1)-49*(0==3) =$$

$$= -3\cdot 1 - 1\cdot 0 - 49\cdot 0 = -3;\ 3-3=0,$$

т. е. первым символом зашифрованного сообщения будет «0»;
сдвиг для второго символа:

$$-3*(1==0)-1*(1==1)-49*(1==3) =$$

$$= -3\cdot 0 - 1\cdot 1 - 49\cdot 0 = -1;$$

сдвиг для третьего символа:

$$-3*(2==0)-1*(2==1)-49*(2==3) =$$

$$= -3\cdot 0 - 1\cdot 0 - 49\cdot 0 = 0;$$

сдвиг для четвертого символа:

$$-3*(3==0)-1*(3==1)-49*(3==3) =$$

$$= -3\cdot 0 - 1\cdot 0 - 49\cdot 1 = -49.$$

Стохастический разброс получаемых значений в результате применения различных функций приведен в таблице.

Результаты использования математических выражений для определения сдвига символов сообщения при его зашифровании

Формула	Номер символа в сообщении									
	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
x/5	0	0	0	0	1	1	1	1	1	2
sumDel(x)	1	2	3	6	5	11	7	14	12	17
sumDelM(x)	0	0	0	2	0	5	0	6	3	7
sin(x)*10	8	9	1	–7	–9	–2	6	9	4	–5
delModuleD(x; 1.75)	1	0	1	1	1	0	2	1	0	0
max(x;3)%min(x;3)	0	1	0	1	2	0	1	2	0	1
x++ + x	3	5	7	9	11	13	15	17	19	21

Преимуществами авторского метода зашифрования/расшифрования сообщений на основе ключа в виде рекуррентных соот-

ношений по сравнению с подобными методами, основанными на использовании статического ключа, являются:

- устойчивость к большим объемам данных;
- устойчивость к часто повторяющимся данным;
- специфика зашифрования, позволяющая создавать трудно анализируемые выражения.

Недостатками же является:

- зависимость криптостойкости ключа от составленной формулы, что ограничивает круг пользователей, способных самостоятельно создать сложные ключи;
- более медленная обработка по сравнение с прямым гаммированием;
- сложность анализа результирующей работы ключа.

Для решения проблемы анализа ключа можно использовать модель визуализатора: ключ в виде графа, дугами которого являются математические операции, а узлами – результаты этих операций. Также, помимо графа, визуализацию ключа можно провести с помощью построения матрицы смежности.

14.3.3. Метод на основе преобразования кода в нечитабельные инструкции

Основополагающая идея метода не нова.

Суть метода состоит в преобразовании кода в последовательность нечитабельных инструкций. При этом ключевыми моментами в реализации являются:

- добавление большого количества не влияющих на выполнения программы конструкций, позволяющих решить проблему статического расположения участка кода, производящего выполнение программы;
- использование во время преобразования программы множества тождественных переменных с их произвольным выбором на соответствующих фрагментах кода;
- изменение значений переменных на протяжении всего кода, что не позволяет производить обратное преобразование на основе прямолинейной замены.

Реализована модель составления и хранения обфусцирующих конструкций в отдельных текстовых файлах для последующего использования при генерации кода: «умного мусора», тождественных

либо допустимых по точности математических операций, преобразованных строковых данных, операторов сравнения и присваивания, измененных цикловых конструкций и др. (рис. 14.2).

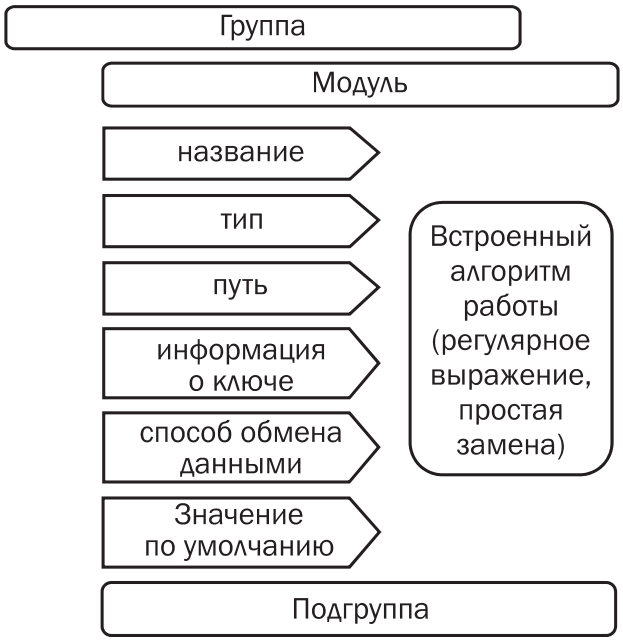


Рис. 14.2. Общая модель хранения шаблонов кода, используемого при генерации обфусцирующих вставок

В качестве источника для заполнения разработанной базы использовался ресурс *wtfjs*, из которого были извлечены 95 неочевидных по выполнению приемов, среди которых есть такие, как работа методов класса *Math* при передаче разнотипных данных; операции над массивами и коллекциями; манипулирование областями видимости объектов при объявлении и обращении к объектам; сравнение различных операций с логическими выражениями; упаковка функций в изображения с асинхронной загрузкой; перекрытие функций известных библиотек и многие другие.

Пример 14.26. Пример конструкции для манипулирования областями видимости объектов выглядит так:

```
var a = 1, // используется запятая
    b = 1;
(function(){
      var a = 2 // НЕ используется запятая
b = 2; // в результате не создается локальная b, а используется
внешняя
```

```
}());
console.log(a); // 1
console.log(b); // 2
```

Для подобной конструкции может быть использован следующий шаблон:

```
var[[NAME1]][=[VALUE1]],
[[NAME2]][=[VALUE2]];
[[CODE1]]
(function(){
    var [[NAME1]]=[[VALUE3]]
[[NAME2]]=[[VALUE4]]
    [[CODE2]]
}());
```

Использование двойных квадратных скобок необходимо для того, чтобы реализовать возможность использования необязательных значений. Так, например «*[=[VALUE1]]*» будет преобразовано в пустую строку, если значение *VALUE1* не указано. Выражение «*=[[VALUE3]]*» будет преобразовано к «*=value3*» в любом случае.

Маркеры *[[CODE1]]* и *[[CODE2]]* позволяют разбавить конструкцию дополнительным кодом для того, чтобы сильнее затруднить анализ. Помимо этого, перед и после любого шаблона можно также указать необходимый код, используя при вызове обработчика необязательные ключи *CODE_BEFORE* и *CODE_AFTER*.

Тождественное математическое преобразование, представленное ранее при описании *тождественных либо допустимых по точности математических операций* для записи чисел, преобразуется в следующий шаблон (хранение заменителя, основанного на математическом тождестве):

```
[[NAME2]] = DataTime.Now;
[[NAME1]] = [[NAME2]]%10+(([[NAME2]]%100)/10)%10)-
([[NAME2]]%100)%9+[[VALUE1]];
```

В этом шаблоне используется дополнительная переменная *NAME2*. Если ее название не будет установлено, то используется значение по умолчанию.

Операции обфускации являются необратимыми. Тем не менее было бы полезным иметь возможность восстановить после их

применения исходный код, сохранив для этого в отдельном файле необходимые данные (ключи). Также полезным представляется использование регулярных выражений для поиска и замены по собственным правилам.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ

1. Дайте сравнительную оценку и поясните сущность известных методов защиты кодов программ от несанкционированного использования и модификации.

2. На чем основана идея методов обфускации?

3. Продемонстрируйте на конкретном примере реализацию выбранного (или указанного преподавателем) метода обфускации программного кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Печатные издания

1. Shannon, C. E. The communication theory of secrecy systems / C. E. Shannon // Bell Sys. Tech. J. – 1949. – Vol. 28, No. 4. – P. 656–715.

2. Шнаейер, Б. Секреты и ложь. Безопасность данных в цифровом мире / Б. Шнаейер. – СПб.: Питер, 2003. – 370 с.

3. Шнаейер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке СИ / Б. Шнаейер. – М.: Триумф, 2003. – 816 с.

4. Онацкий, А. В. Асимметричные методы шифрования. Модуль 2. Криптографические методы защиты информации в телекоммуникационных системах и сетях: учеб. пособие / А. В. Онацкий, Л. Г. Йона; под ред. Н. В. Захарченко – Одесса: ОНАС им. А. С. Попова, 2010. – 148 с.

5. Ярмолик, В. Н. Криптография, стеганография и охрана авторского права: монография / В. Н. Ярмолик, С. С. Портянко, С. В. Ярмолик. – Минск: Изд. центр БГУ, 2007. – 240 с.

6. Аграновский, А. В. Основы компьютерной стеганографии: учеб. пособие для ВУЗов / А. В. Аграновский, И. Н. Хади, А. В. Черемушкин. – М.: Радио и связь, 2003. – 152 с.

7. Грибунин, В. Г. Цифровая стеганография. Аспекты защиты / В. Г. Грибунин, И. Н. Оков, И. В. Туринцев. – М.: Солон-Пресс, 2002. – 272 с.

8. Урбанович, П. П. Защита информации и надежность информационных систем / П. П. Урбанович, Д. В. Шиман. – Минск: БГТУ, 2013. – 90 с.

9. Урбанович, П. П. Информационная безопасность и надежность систем / П. П. Урбанович, Д. М. Романенко, Е. В. Романцевич. – Минск: БГТУ, 2007. – 90 с.

10. Хогланд, Г. Взлом программного обеспечения. Анализ и использование кода / Г. Хогланд, Г. Мак-Гроу. – М.: Издательский дом «Вильямс», 2005. – 400 с.

11. Болотов, А. А. Алгоритмические основы эллиптической криптографии / А. А. Болотов, С. Б. Гашков, А. Б. Фролов. – М.: КУДИЦ-ПРЕСС, 2000. – 214 с.

12. Баричев, С. Криптография без секретов / С. Баричев. – СПб.: БХВ-Петербург, 2003. – 43 с.

13. Ховард, М. Защищенный код / М. Ховард, Д. Лебланк. – 2-е изд., испр. – М.: Издательско-торговый дом «Русская Редакция», 2005. – 704 с.

14. Urbanovich, N. The use of steganographic techniques for protection of intellectual property rights / N. Urbanovich, V. Plaskovitsky // Proc. of the 7-th Intern. Conf. on New Electrical and Electronic Technologies and their Industrial Implementation, Zakopane, Poland, 28.06–01.07.2011. – Zakopane, 2011. – P. 147–148.

15. Urbanovich, N. Development, analysis of efficiency and performance in an electronic textbook methods of text steganography / N. Urbanovich // Printing future days: 4th International Scientific Conference on Printing and Media Technology. – Chemnitz, Germany, 07–10.11.2011. – Chemnitz, 2011. – P. 189–193.

16. Пласковицкий, В. А. Защита программного обеспечения от несанкционированного использования и модификации методами обфускации / В. А. Пласковицкий, П. П. Урбанович // Труды БГТУ. – 2011. – № 6: Физ.-мат. науки и информатика. – С. 173–176.

17. Пласковицкий, В. А. Шифрование кодов программ на основе ключа, задаваемого рекуррентными математическими соотношениями / В. А. Пласковицкий, П. П. Урбанович // Труды БГТУ. – 2012. – № 6: Физ.-мат. науки и информатика – С. 146–148.

18. Text steganography application for protection and transfer of the information / P. Urbanovich [et al.] // Przeglad elektrotechniczny. – 2012. – № 8. – P. 342–344.

19. Шутько, Н. П. Особенности и формальное описание процесса осаждения секретной информации в текстовые документы на основе стеганографии / Н. П. Шутько // Труды БГТУ. – 2014. – № 6: Физ.-мат. науки и информатика. – С. 121–124.

20. Пласковицкий, В. А. Использование абстракных синтаксических деревьев для обфускации кода / В. А. Пласковицкий, П. П. Урбанович // Труды БГТУ. – 2014. – № 6: Физ. мат. науки и информатика. – С. 142–146.

21. Шутько, Н. П. Математическая модель системы текстовой стеганографии на основе модификации пространственных и цветовых параметров символов текста / Н. П. Шутько, Д. М. Романенко, П. П. Урбанович // Труды БГТУ. – 2015. – № 6: Физ.-мат. науки и информатика. – С. 152–157.

Интернет-ресурсы

1. Keith, M. J. Quick defense personal self-defense system / M. J. Keith // Protection, security and safeguards [Электронный ресурс]. – 2011. – Режим доступа: https://www.javelinstrategy.com/Brochure-209. – Дата доступа: 16.09.2015.

2. Охрана авторских прав в сети Интернет // Сайт БарГУ [Электронный ресурс]. – 2011–2016. – Режим доступа: http://bargu.by/658-oxrana-avtorskix-prav-v-seti-internet.html. – Дата доступа: 21.11.2015.

3. Intelligence, surveillance and privacy // CSIS [Электронный ресурс]. – 2015. – Режим доступа: http://csis.org/.../2014-mcafee-report-global-cost-... . – Дата доступа: 13.11.2015.

4. Creation of a global culture of cybersecurity and taking stock of national efforts to protect critical information infrastructures // Miltilingual interface of the UN Official Documents System [Электронный ресурс]. – 2015. – Режим доступа: http://daccess-dds-ny.un.org/doc/UNDOC/GEN/N09 /474/49/PDF/N0947449.pdf. – Дата доступа: 13.02.2015.

5. The International Obfuscated C Code Contest // IOCCC [Электронный ресурс]. – 1984–2016. – Режим доступа: http://www.ioccc.org/. – Дата доступа: 07.05.2014.

6. О технологии ClicOnce // Inquartos Obfuscator. Защита .NET-приложений [Электронный ресурс]. – 2008. – Режим доступа: http://netobf.com/ obf_click_once_ about. – Дата доступа: 11.05.2014.

7. Ключевое слово в защите информации // Интернет-портал компании Крипто-про [Электронный ресурс]. – 2000–2016. – Режим доступа: http://www.cryptopro.ru. – Дата доступа: 10.05.2013.

8. Dean Packer // Internet-portal Dean Edwards [Электронный ресурс]. – 2004–2016. – Режим доступа: http://dean.edwards.name/download/#packer. – Дата доступа: 15.05.2014.

9. Tufte, E. R. Plagiarism detection in PowerPoint presentations / E. R. Tufte [Электронный ресурс]. – 2016. – Режим доступа: http://www.edwardtufte.com/tufte/. – Дата доступа: 15.05.2014.

10. Обфускация и защита программных продуктов // CIT Forum [Электронный ресурс]. – 2001–2015. – Режим доступа: http://citforum.ru/security/articles/obfus/. – Дата доступа: 25.12.2015.

11. Эллиптическая кривая // Википедия: свободная энциклопедия [Электронный ресурс]. – 2015. – Режим доступа: https://ru.wikipedia.org/wiki/Эллиптическая кривая. – Дата доступа: 14.06.2015.

12. Эллиптическая криптография: теория // Интернет-сайт Хабрахабр [Электронный ресурс]. – 2013. – Режим доступа: https:// habrahabr.ru/ post/188958/. – Дата доступа: 14.06.2015.

13. Потоковый шифр // Википедия: свободная энциклопедия [Электронный ресурс]. – 2015. – Режим доступа: https://ru.wikipedia.org/wiki/Потоковый шифр. – Дата доступа: 19.07.2015.

14. Шифры простой замены // Сайт Казанского государственного технического университета [Электронный ресурс]. – 2002. – Режим доступа: http://crypto-r.narod.ru/glava2/glava2_3.html. – Дата доступа: 19.01.2016.

ОГЛАВЛЕНИЕ

Учебное издание

Урбанович Павел Павлович

ЗАЩИТА ИНФОРМАЦИИ МЕТОДАМИ КРИПТОГРАФИИ, СТЕГАНОГРАФИИ И ОБФУСКАЦИИ

Учебно-методическое пособие

Редактор *О. П. Приходько*
Компьютерная верстка *О. Ю. Шантарович*
Корректор *О. П. Приходько*

Подписано в печать 22.09.2016 . Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Гарнитура Таймс. Печать офсетная.
Усл. печ. л. 12,8. Уч.-изд. л. 13,2.
Тираж 100 экз. Заказ .

Издатель и полиграфическое исполнение:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/127 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.