Министерство образования Республики Беларусь

П.П. Урбанович

**Защита информации методами криптографии, стеганографии и обфускации**

Минск 2016

1

УДК 003.26+346.78

ББК 32.811+67.404.3

Ухх

Рецензенты:

кафедра электронных вычислительных средств УО «Белорусский государст-

венный университет информатики и радиоэлектроники (заведующий кафед-

рой Петровский А.А., профессор, д.т.н.);

Листопад Н.И., директор ГУ «Главный информационно-аналитический центр

Министерства образования Республики Беларусь) , профессор, д.т.н

*Все права на данное издание защищены. Воспроизведение всей книги или ее части не может быть осуществлено без разрешения учреждения образова-ния «Белорусский государственный технологический университет».*

**Урбанович, П.П.**

Защита информации методами криптографии, стеганографии и обфуска-ции: учеб.-метод. пособие для студентов специальностей: 1-40 05 01-03 «Ин-формационные системы и технологии (издательско-полиграфический ком-плекс)», 1-40 01 01 «Программное обеспечение информационных техноло-

гий» (специализация 1-40 01 01-10 «Программирование интернет-приложений»), 1-47 01 02 «Дизайн электронных и веб-изданий», 1-98 01 03 «Программное обеспечение информационной безопасности мобильных сис-тем»/ П.П. Урбанович. – Минск: БГТУ, 2016. - с.

ISSBN 978-985-530-

Изложены основные понятия, относящиеся к проблеме защиты информации в ком-пьютерных 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, защи-ты права интеллектуальной собственности на текстовые документы, базы данных и коды программ, генерации электронной цифровой подписи и ее проверки.

Пособие предназначено для студентов и магистрантов ИТ-специальностей, изучаю-щих дисциплины, относящиеся к указанной предметной области. Может быть полезно на-учным работникам, объектами исследований которых являются методы и средства обес-печения информационной безопасности систем и технологий.

**УДК 003.26+346.78**

**ББК 32.811+67.404.3**

ISSBN 978-985-530-

© УО «Белорусский государственный технологический университет», 2016 © Урбанович П.П., 2016

2

**ВВЕДЕНИЕ**

За последние 10-15 лет информационные технологии существенно расширили и усилили свой плацдарм во всех сферах нашей жизнедеятельно-сти. Это обстоятельство – безусловный положительный фактор, влияющий на инновационный характер развития реального сектора экономики, здраво-охранения, сферы услуг, досуга и, конечно же, образования. Однако, наряду с этим, указанный тренд со все большей очевидностью обнажает остроту про-блем, негативных последствий информатизации. В наибольшей степени эти проблемы связаны с возможностями несанкционированного доступа к ин-формационным ресурсам, объектам инфраструктуры, принадлежащим дру-гим физическим лицам, субъектам хозяйствования, банковской сферы, дру-гим государствам. Это напрямую связано не только с необходимостью обес-печения информационной, но и государственной безопасности.

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

* рамках существующих стандартов образования и учебных планов ряда специальностей (в том числе – 1-40 05 01-03 «Информационные систе-мы и технологии (издательско-полиграфический комплекс)», 1-40 01 01 «Программное обеспечение информационных технологий» (специализация 1-40 01 01-10 «Программирование интернет-приложений»), 1-47 01 02 «Ди-зайн электронных и веб-изданий», 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем») предусматривается изу-чение студентами комплекса специальных дисциплин, направленных на ов-ладение теоретическими знаниями и практическими навыками в анализируе-мой предметной области. К числу таких дисциплин относятся: Защита ин-формации и надежность информационных систем (специальность 1-40 05 01-03), Криптографические методы защиты информации (1-98 01 03, 1-40 01 01-10, 1-47 01 02).

С учетом того обстоятельства, что динамика процессов в ИТ-сфере требует адекватного реагирования путем внесения соответствующих измене-ний в учебные планы, подготовки учебно-методических пособий, следует признать, что методическое обеспечение учебных дисциплин, относящихся к информационной безопасности, в ВУЗах пока не носит системного характе-ра. Данное пособие, по мнению авторов, призвано восполнить указанный пробел.

Основное содержание книги охватывает рассмотрение вопросов, от-носящихся к трем основным классам методов преобразования информации с целью ее защиты от несанкционированного доступа и использования, нару-

3

шения целостности, а также защиты права интеллектуальной собственности:

криптографии, стеганографии и обфускации.

Материал пособия излагается в последовательности, которая, по на-шему мнению, является оптимальной с методической точки зрения при изу-чения его в полном объеме. Хотя, конечно, данная точка зрения не претенду-ет на бесспорность. С другой стороны, мы старались оформить каждую главу

* определенной степени автономной, что может повысить эффективность изучения избранных глав.

После разделов, а в некоторых случаях и после подразделов даны во-просы для самостоятельного контроля знаний либо контроля знаний обучае-мого преподавателем. На основе сформулированных вопросов и заданий мо-гут быть составлены тесты для компьютерного (дистанционного) контроля знаний. Наш опыт показывает, что такая форма контроля достаточно эффек-тивна не только для студентов, обучающихся по заочной (или дистанцион-ной) форме образования, но и студентов-дневников.

Глава 3 пособия написана автором совместно с Н.П. Шутько, глава 5 – совместно с В.А. Пласковицким.

Книга рассчитана на студентов, изучающих перечисленные выше и родственные дисциплины; может быть полезной магистрантам ИТ-специальностей, научным работникам, объектами исследований которых яв-ляются методы и средства обеспечения информационной безопасности сис-тем и технологий.

4

**ГЛАВА I МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ И ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ СИСТЕМ**

**1. Фундаментальные понятия и определения из области инфор-мационной безопасности систем**

***1.1 Краткая историческая справка***

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

**I этап** –примерно до середины15века,когда И.Гутенберг(JohannesGutenberg) создал европейский способ книгопечатания подвижными литера-ми, распространившийся по всему миру. До этого времени практически лю-бая информация передавалась устно, т.е конфиденциальность определялась исключительно «человеческим фактором».

* 1. **этап** –примерно до начала19века–характеризуется использова-нием естественно возникавших [средств информационных коммуникаций](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Основная задача информационной безопасности – защита сведений о собы-тиях, фактах, имуществе и т. д.

**III этап** –связан с началом использования технических средств [элек-тро-](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%81%D0%B2%D1%8F%D0%B7%D1%8C) и [радиосвязи](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%BE%D1%81%D0%B2%D1%8F%D0%B7%D1%8C). Характеризуется применением помехоустойчивого [коди-рования](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [сообщения](http://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5) ([сигнала](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB)) с последующим декодированием принятого со-общения (сигнала).

**IV этап** –начиная с1935года–связан с появлением [радиолокацион-ных и гидроакустических средств](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%BE%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0). Обеспечение информационной безопасно-сти основывалось на сочетании организационных и технических мер, на-правленных на повышение защищенности радиолокационных средств от воздействия на их приемные устройства активных и пассивных помех.

**V этап** –начиная с1946года–связан с изобретением и внедрением впрактическую деятельность [электронно-вычислительных машин](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) (компьюте-ров). Эру появления компьютерной техники связывают с разработкой в Пен-сильванском университете (США) ЭВМ EN IAC (*Electronic Numerical Inte-grator And Computer (Calculator)*).Задачи информационной безопасности ре-шались в основном методами и способами ограничения физического доступа

* оборудованию средств сбора, переработки и передачи информации.

**VI этап** –начиная с первой половины60-х годов20века–обуслов-лен созданием и развитием [локальных](http://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) [информационно-коммуникационных](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) [сетей](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Задачи безопасности решались в основном методами и способами фи-зической защиты средств, путем [администрирования](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и управления доступом к сетевым ресурсам.

**VII этап** –начиная с1973года–связан с использованием мобильныхкоммуникационных устройств с широким спектром задач. В этот период соз-даны известные сейчас во всем мире фирмы Microsoft (Билл Гэйтс и Пол

5

Ален) и Apple (Стив Джобс и Стэфан Возняк).

Образовались сообщества людей – [*хакеров*](http://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%BA%D0%B5%D1%80), ставящих своей целью на-несение ущерба информационной безопасности отдельных пользователей, организаций и целых стран. Формируется [*информационное право*](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE) – новая от-расль международной правовой системы.

**VIII этап** – начиная примерно с 1985 года – связан с созданием и раз-витием [глобальных](http://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) [информационно-коммуникационных сетей](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) с использова-нием космических средств обеспечения.

Предусматривает комплексное использование мер и средств защиты. **IХ этап** – примерно с концаXX– началаXXIвв. – связан с повсеме-

стным использованием сверхмобильных коммуникационных устройств с широким спектром задач и глобальным охватом в пространстве и времени, обеспечиваемым космическими информационно-коммуникационными сис-темами. Характеризуется *«широким переходом на цифру»*, использованием *облачных технологий*, хранением, обработкой и использованием большихобъемов данных (*Big Data*)*.* Предусматривает комплексное использование мер и средств защиты.

Проблема защиты информации путем ее преобразования относится к [*криптологи*](http://ru.science.wikia.com/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F)и([греч.](http://ru.science.wikia.com/wiki/%D0%93%D1%80%D0%B5%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)κρυπτός- *kryptos* — тайный,λόγος *- logos* — слово).Криптология разделяется на два направления — криптографию и [криптоана-лиз.](http://ru.science.wikia.com/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) Цели этих направлений прямо противоположны.

*Криптография* (греч.κρυπτόςиγράφω- grafo— пишу) или тайно-пись занимается поиском и исследованием математических методов преобра-зования информации. Сфера интересов *криптоанализа* — исследование воз-можности расшифровывания информации или взлома шифров.

Криптография — одна из старейших наук, ее история насчитывает не-сколько тысяч лет. Первоначально письменность сама по себе была крипто-графической системой, так как в древних обществах ею владели только из-бранные. Священные книги Древнего Египта, Древней Индии тому примеры.

* широким распространением письменности криптография стала формиро-ваться как самостоятельная наука. Первые криптосистемы встречаются уже в начале нашей эры. Бурное развитие криптографические системы получили в годы первой и второй мировых войн.
  + конце 1918 года в свет вышел один из самых значительных трудов двадцатого столетия в области криптоанализа - монография Уильяма Ф. Фридмана (William F. Friedman) *«1пdex of Coincidense and its Application in* *Cryptograph*у» (Индекс совпадения и его применение к криптографии)1. При-мерно в то же время Эллард X. Хеберн (США) получил первый патент (US Patent №1510441) на роторную машину — устройство, которому суждено было стать основой военной криптографии почти на полвека.

1. Переиздано: The Index of Coincidence and Its Application in Cryptography, Laguna Hills, CA: Aegean Park Press, 1986

6

На протяжении 30-40-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 «*The Communication Theory of Secrecy Systems*»2 (Теория связи в сек-ретных системах), которая послужила началом обширных исследований в теории кодирования и передачи информации, и, по всеобщему мнению, при-дала криптографии статус науки.

* конце 60-х-начале 70-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).
* 1975 году У. Диффи (W. Diffie) и М. Хеллман (M.Hellman) предло-жили новый метод криптографического преобразования информации – крип-тографию с открытым ключом или асимметричную криптографию3.

Последние десятилетия основным объектом исследований криптогра-фической науки стала не теория, а практика. Это обусловлено несколькими причинами. Основной среди них, видимо, является развитие интернет-технологий.

***1.2 Основные понятия и определения из области защиты информации***

Далее сформулируем определения основных понятий, которыми бу-дем пользоваться при изучении предметной области.

*Информация* –сведения(данные)о внутреннем и окружающем насмире, событиях, процессах, явлениях и т. д., воспринимаемые и передавае-мые людьми или техническими устройствами.

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

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

*Информационные ресурсы* –отдельные документы или массивы до-кументов в информационных системах.

*Объект –* пассивный компонент системы,хранящий,перерабатываю-щий, передающий или принимающий информацию; примеры объектов: стра-ницы, файлы, папки, директории, компьютерные программы, устройства (мо-ниторы, диски, принтеры и т. д.).

*Субъект –* активный компонент системы,который может иницииро-

1. Shannon, C. Communication Theory of Secrecy Systems/ Bell System Technical Journal, 1949, v. 28(**4**), р. 656–
2. W. Diffie and M. E. Hellman. New Directions in Cryptography/ IEEE Transactions on Information Theory, 1976, v. IT-22, Nov., p. 644—654

7

вать поток информации; примеры субъектов: пользователь, процесс либо уст-ройство.

*Доступ –* специальный тип взаимодействия между объектом и субъ-ектом, в результате которого создается поток информации от одного к дру-гому.

*Несанкционированный доступ (НСД) –* доступ к информации,устрой-ствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные пра-вила доступа.

*Безопасность ИВС –* свойство системы,выражающееся в способностисистемы 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 безо-пасности и устранению их последствий.

*Атака* –попытка несанкционированного преодоления защиты систе-

мы.

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

Другое определение:

*информационная безопасность –* все аспекты,связанные с определе-нием, достижением и поддержанием конфиденциальности, целостности, дос-тупности информации или средств ее обработки:

[*конфиденциальность*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) *(confidentiality) –* состояние информации,прикотором доступ к ней осуществляют только субъекты, имеющие на нее пра-во;

[*целостность*](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) *(integrity) –* избежание несанкционированной модифи-кации информации;

[*доступность*](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BD%D0%BE%D1%81%D1%82%D1%8C) *(availability) –* избежание временного или постоянногосокрытия информации от пользователей, получивших права доступа*.*

*Идентификация* –процесс распознавания определенных компонентовсистемы (объектов или субъектов) с помощью уникальных идентификаторов.

*Аутентификация* –проверка идентификации пользователя или иногокомпонента ИС для принятия решения о разрешении доступа к ресурсам систе-мы.

*Надежность системы* –характеристика способности программного,аппаратного, аппаратно-программного средства выполнить при определен-ных условиях требуемые функции в течение определенного периода време-ни.

*Ошибка* устройства–неправильное значение сигнала(бита–в циф-ровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или воздействующими на него по-

8

мехами (преднамеренными либо непреднамеренными), или иным способом. *Ошибка* программы–проявляется в не соответствующем реальному

(требуемому) промежуточному или конечному значению (результату) вслед-ствие неправильно запрограммированного алгоритма или неправильно со-ставленной программы.

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

*Фактор*, *воздействующий на ИВС*,–это явление,действие или про-цесс, результатом которых может быть утечка, искажение, уничтожение дан-ных, блокировка доступа к ним, повреждение или уничтожение системы за-щиты.

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

*Внутренние дестабилизирующие факторы*,влияющие:

1. на программные средства (ПС):

а) некорректный исходный алгоритм;

б) неправильно запрограммированный исходный алгоритм (первич-ные ошибки);

1. 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 состоя-нием АС.

*Внешние дестабилизирующие факторы*,влияющие:

1. на программные средства:

а) неквалифицированные пользователи;

б) несанкционированный доступ к ПС с целью модификации кода;

1. 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. Дать определение основных понятий и терминов, относящихся к области защиты информации и надежности информационных систем.
2. Чем отличается идентификация от авторизации?
3. Охарактеризовать основные этапы развития информационных технологий с точки зрения их безопасности.

9

4. Привести классификацию основных факторов, влияющих на

ИВС.

1. К каким последствиям приводит влияние дестабилизирующих факторов на ИС или ИВС?
2. Дать характеристику внутренних факторов, дестабилизирующих работу ИС или ИВС.
3. Охарактеризовать внешние факторы, дестабилизирующие работу ИС или ИВС.
4. Как Вы понимаете «некорректный исходный алгоритм»?
5. Что такое «первичная ошибка» в программе?
6. Как влияют климатические условия на надежность аппаратных

средств?

10

1. **Потенциальные угрозы безопасности информации в ИВС. Объекты и методы защиты информации**

***2.1 Естественные и искусственные помехи и угрозы безопасности***

Одним из важнейших дестабилизирующих работу ИВС факторов яв-ляются *электромагнитные* и *ионизирующие* излучения. Источниками первых являются практически все средства, функционирование которых основано на использовании электроэнергии, и в особенности такие АС, которые целена-правленно излучают электромагнитные волны (к ним относится, например, приемо-передающие и иные подобные радиоэлектронные устройства). По большому счету любой проводник с током является источником электромаг-нитных помех. Такие источники относятся к классу *искусственных* или *про-мышленно-бытовых*.В свою очередь,их можно подразделить на *непреднаме-ренные* и *преднамеренные*.Последние имеют место в ситуациях,похожих наэпизод в фильме о приключениях Шурика: профессор на экзамене включил генератор помех, который «забил» канал.

*Ионизирующие* излучения также могут иметь естественную(солнеч-ная радиация) и искусственную (изотопы урана и тория излучают даже пла-стмассы) природу.

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

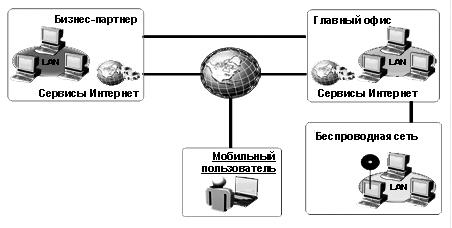


Рис. 2.1. Пример периметра современной ИС

Для лучшего понимания сути и особенностей *угроз со стороны дест-руктивных* или *злонамеренных программных средств* (англ*. malware* —***ma-l****icious soft****ware***)*,*а также физических лиц,использующих такие средства дляорганизации НСД (хакеры и кракеры), на рис. 2.1 приведен пример перимет-ра современной ИС, а на рис. 2.2 схематически показаны наиболее уязвимые

11

места локальной сети.

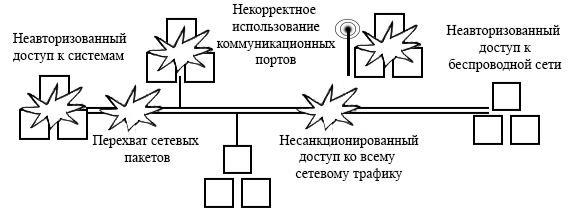


Рис. 2.2. Наиболее уязвимые места локальной сети

*Основные факторы (угрозы):*

1. действия злоумышленника;
2. наблюдение за источниками информации;
3. подслушивание конфиденциальных разговоров и акустических сиг-налов 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 информа-

ции;

1. отказы системы электроснабжения;
2. воздействие мощных электромагнитных и электрических помех (промышленных и природных).

Несанкционированный доступ с помощью *деструктивных программ-ных средств* осуществляется,как правило,через компьютерные сети.

*Классификацию вредоносного ПО* можно представить следующим обра-

зом:

*вирусы* (viruses)–этосаморазмножающиеся программыпутем допи-сывания собственных кодов к исполняемым файлам; вирусы могут содер-жать, а могут не содержать деструктивные функции;

*черви* (worms)–это программы,которые самостоятельно размножа-ются по сети и, в отличие от вирусов, не дописывают себя (как правило) к исполняемым файлам; все черви «съедают» ресурсы компьютера, «нагоня-

12

ют» интернет-трафик и могут привести к утечке данных с вашего компьюте-ра;

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

*трояны* (trojans),или *троянские кони*,–собирают конфиденциальнуюинформацию 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 ис-пользуется для атак на другие компьютеры;

*снифферы* (sniffers)–это анализаторы сетевого трафика;могут ис-пользоваться в составе зловредного ПО, скрытно устанавливаться на компь-ютере пользователя и отслеживать данные, которые отправляет или получает пользователь по сети;

*руткиты* (rootkits)–сами по себе не являются зловредным ПО;назна-чение – 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

* пр.).

Важно отметить, что вирусы, трояны и иные деструктивные програм-мы активизируются только после загрузки инфицированного файла в опера-тивную 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 поль-зователем определенных действий, например при неосмотрительном запуске неизвестного файла. Много проблем могут доставить полиморфные вирусы: их программный код постоянно меняется, что делает их обнаружение клас-сическими 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 про-никновения на ПК действуют самостоятельно, так как являются автономны-ми программами. В категорию троянских коней входят программы для хи-щения 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 дру-гими видами сетевых угроз, подключаясь к Интернету через бесплатные точ-

13

ки доступа Wi-Fi, например, для проверки банковского счета, оплаты мо-бильного 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 ле-гальные бесплатные точки доступа;

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

*сниффинг* (сниффер,от англ. *to sniff —* нюхать)—сетевой анализатортрафика, программа или программно-аппаратное устройство, предназначен-ное 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 мо-жет читать сообщения, которыми обмениваются пользователи так, чтобы они даже не догадывалась о его присутствии.

* + 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 ис-пользовать только проверенные точки доступа к сети Интернет. Для доступа
* 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 доступа к конфиденци-альным данным пользователей — логинам и паролям. Это достигается путѐм 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 раз-личными психологическими приѐмами побудить пользователя ввести на поддельной странице свои логин и пароль, которые он использует для досту-па к определѐнному сайту, что позволяет мошенникам получить доступ к банковским счетам.

По данным известной Лаборатории Касперского к числу стран, наи-более часто подвергающихся в 2014 году атакам банковскими троянцами, от-носятся, в основном, страны бывшего СССР. По тем же аналитическим дан-ным карта попыток заражений мобильными зловредными программами, по-лученная по 213 странам мира, использующим антивирусное ПО Лаборато-

14

рии Касперского, выглядит так, как представлено в таблице 2.1 (процент от всех атакованных уникальных пользователей).

Таблица 2.1 Карта выявленных попыток заражений ИС мобильными зловредными программами

|  |  |
| --- | --- |
| Страна | % атак |
|  |  |
| Россия | 48,90% |
|  |  |
| 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 уязвимы с точки зрения защищенности ресурсов являются так называемые критические информационные системы.

*Критическая И*С(КИС) –это сложная компьютеризированная орга-низационно-техническая система, блокировка или нарушение функциониро-вания которой потенциально приводит к потере устойчивости организацион-ных систем государственного управления и контроля, утрате обороноспособ-ности государства, разрушению системы финансового обращения, дезорга-низации систем энергетического и коммуникационно- транспортного обеспе-чения государства, глобальным экологическим или техногенным катастро-фам.

При решении проблемы повышения уровня защищенности информа-ционных ресурсов КИС необходимо исходить из того, что наиболее вероят-ным информационным объектом воздействия будет выступать программное обеспечение, составляющее основу комплекса средств получения, семанти-ческой переработки, распределения и хранения данных, используемых при эксплуатации таких систем.

При рассмотрении правовых аспектов защиты информации и инфор-мационной безопасности противоправные действия классифицируются как *компьютерное преступление*.Наказания за преступления против информа-ционной безопасности впервые были юридически закреплены в Уголовном кодексе Беларуси в 1999г.

* + *уголовном праве Беларуси* закреплена ответственность за ряд пре-ступлений против информационной безопасности (Глава 31 УК РБ). К пре-ступлениям, связанным с использованием компьютерных средств (подлог и мошенничество, совершенные с использованием компьютерных технологий)
* Уголовном Кодексе Республики Беларусь относят хищение путем исполь-

15

зования компьютерной техники (ст. 212); причинение имущественного ущерба без признаков хищения (ст. 216) путем модификации компьютерной информации); ряд составов, которые включают хищение (ст. 294, 323, 327, 333). Третья группа преступлений включает преступления, связанные с со-держанием компьютерных данных: детская порнография, нарушение автор-ского права.

* отношении нарушений *авторских прав* с использованием компью-терных технологий уголовное право Республики Беларусь не предусматрива-ет каких-либо специальных норм. Некоторые из рассмотренных составов преступлений имеют свои аналоги в *Кодексе об административных право-*

*нарушениях* (ст22.6КоАП РБ–несанкционированный доступ к компьютер-ной информации; cт. 10.7 КоАП РБ – причинение имущественного ущерба)4.

* целом перечень преступлений против информационной безопасно-сти, 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 *ки-берпреступности*,включая незаконный доступ к компьютерным системам иперехват данных, воздействие на данные, воздействие на работу системы, противозаконное использование устройств, подлог и мошенничество с ис-пользованием компьютерных технологий, правонарушения, связанные с дет-ской порнографией, и правонарушения, связанные с авторским правом и смежными правами. При подготовке конвенции преследовались цели форми-рования общей правоохранительной системы для борьбы с киберпреступно-стью и создания условий для обмена информацией между всеми странами, подписавшими конвенцию. В Будапештской конвенции также устанавлива-ются требования защиты прав и свободы каждого в сети интернет.

Международная уголовная полиция "Интерпол" пользуется *классифи-*

*кацией компьютерных преступлений* по кодификатору международной уго-

ловной полиции генерального секретариата Интерпола. В 1991 году данный кодификатор был интегрирован в автоматизированную систему поиска и в настоящее время доступен подразделениям Национальных центральных бю-ро Международной уголовной полиции "Интерпол" более чем 120 стран ми-ра.

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:

QA – Несанкционированный доступ и перехват: QAH –компьютерный абордаж,

QAI – перехват,

QAТ –кража времени,

1. http://www.lawtrend.org/information-access/blog-information-access/informatsionnaya-bezopasnost-prestupleniya-i-nakazaniya (доступ 10.02.2015)
2. http://conventions.coe.int/treaty/rus/Treaties/Html/185.htm (доступ 10.02.2015)

16

QAZ – прочие виды несанкционированного доступа и перехвата,

QD **–** Использование деструктивных программных средств:

QDL – логическая бомба,

QDT – троянский конь,

QDV – компьютерный вирус,

QDW – компьютерный червь,

QDZ – прочие виды.

QF – Компьютерное мошенничество:

QFC – мошенничество с банкоматами,

QFF - компьютерная подделка,

QFG - мошенничество с игровыми автоматами,

QFM - манипуляции с программами ввода вывода,

QFP - мошенничества с платежными средствами,

QFT - телефонное мошенничество,

QFZ - прочие компьютерные мошенничества.

QR – Незаконное копирование (пиратство):

QRG - компьютерные игры,

QRS - прочее программное обеспечение,

QRT - топография полупроводниковых изделий

QRZ - прочее незаконное копирование.

QS – Компьютерный саботаж:

QSH - с аппаратным обеспечением,

QSS - с программным обеспечением,

QSZ - прочие виды саботажа.

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***

* контексте сформулированной цели здесь кратко проанализируем методы и средства повышения информационной безопасности систем. Во-просы надежности будут рассмотрены во второй части (при подготовке ко второму тесту).

Политика информационной безопасности систем, как и во всех по-добных случаях, должна строиться на основе *системного подхода*, преду-сматривающего всесторонний анализ причин и угроз безопасности, оценки

17

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 (поли-тика информационной безопасности);
* аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ6.

**Законодательная и нормативно-правовая база**.

1. Акты национального законодательства: а) международные договоры РБ; б) Конституция РБ;

в) законы РБ, например, [*Закон Республики Беларусь от 10 ноября*](http://zakon.by/webnpa/text.asp?RN=h10800455)

[*2008 г. № 455-3* «Об информации,информатизации и защите информации»](http://zakon.by/webnpa/text.asp?RN=h10800455);

г) Указы Президента РБ, например, [*Указ № 515*](http://www.pravo.by/webnpa/text.asp?RN=P31000515) Президента *Респуб-лики Беларусь от 30 сентября 2010 г.* «О некоторых мерах по развитию сетипередачи данных в Республике Беларусь», *Указ № 60 Президента Республики* *Беларусь от 1 февраля 2010 г.* «О мерах по совершенствованию использованиянационального сегмента сети Интернет»;

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 Бела-русь», где, в частности, дана характеристика внутренним источникам угроз на-циональной безопасности, к числу которых относятся:

* распространение недостоверной или умышленно искаженной ин-формации, способной причинить ущерб национальным интересам Республики Беларусь;
* зависимость Республики Беларусь от импорта информационных технологий, средств информатизации и защиты информации, неконтролируе-мое их использование в системах, отказ или разрушение которых может причи-нить ущерб национальной безопасности;
* несоответствие качества национального контента мировому уров-

ню;

* недостаточное развитие государственной системы регулирования процесса внедрения и использования информационных технологий;
* рост преступности с использованием информационно-коммуникационных технологий;
* недостаточная эффективность информационного обеспечения госу-дарственной политики;
* несовершенство системы обеспечения безопасности критически важных объектов информатизации;

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

18

некоторые внешние источники угроз национальной безопасности РБ:

* открытость и уязвимость информационного пространства Респуб-лики Беларусь от внешнего воздействия;
* доминирование ведущих зарубежных государств в мировом ин-формационном пространстве, монополизация ключевых сегментов информаци-онных рынков зарубежными информационными структурами;
* информационная деятельность зарубежных государств, междуна-родных и иных организаций, отдельных лиц, наносящая ущерб национальным интересам Республики Беларусь, целенаправленное формирование информаци-онных 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 Рес-публики Беларусь за рубежом;
* широкое распространение в мировом информационном простран-стве образцов массовой культуры, противоречащих общечеловеческим и на-циональным духовно-нравственным ценностям;
* попытки несанкционированного доступа извне к информационным ресурсам Республики Беларусь, приводящие к причинению ущерба ее нацио-нальным интересам;

*Указ президента Республики Беларусь* №46 от 23.01.2014 «Об исполь-зовании государственными органами и иными государственными организа-циями телекоммуникационных технологий»;

д) постановления Правительства РБ, например, [*Постановление Со-вета Министров Республики Беларусь от 11 февраля 2006 г. № 192* «Об ут-верждении Положения о сопровождении интернет-сайтов республиканских](http://zakon2006.by.ru/part01/doc10036.shtm) [органов государственного управления, иных государственных организаций,](http://zakon2006.by.ru/part01/doc10036.shtm) [подчиненных Правительству Республики Беларусь»](http://zakon2006.by.ru/part01/doc10036.shtm); [*Постановление Совета*](http://www.pravo.by/webnpa/text.asp?start=1&RN=C21101084) [*Министров Республики Беларусь от 11 августа 2011 г. № 1084*](http://www.pravo.by/webnpa/text.asp?start=1&RN=C21101084) «Овнесенииизменений и дополнений в Постановление Совета Министров Республики Беларусь от 29 апреля 2010 г. № 644», [*Постановление*](http://belgie.by/files/info/Postanovlenie675.doc) Совета *Министров* *Республики Беларусь от 26 мая 2009 г. № 675* «О некоторых вопросах защи-ты информации», [*Постановление*](http://belgie.by/files/info/Postanovlenie673.doc) Совета *Министров Республики Беларусь* *от 26 мая 2009 г. № 673 «*О некоторых мерах по реализации Закона Респуб-лики 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 № 4/11*](http://belgie.by/files/info/Postanovlenie4_11.rtf) *Оперативно-аналитического центра при Президенте Республики Беларусь и Министерства связи и информатизации Республики Беларусь от 29 июня 2010 г.* «Об утверждении положения о порядке ограни-чения доступа пользователей интернет-услуг к информации, запрещенной

19

* распространению в соответствии с законодательными актами», [*Приказ*](http://belgie.by/files/info/PrikazOAC60.doc) [*№ 60*](http://belgie.by/files/info/PrikazOAC60.doc) *Оперативно-аналитического центра при Президенте Республики Бела-русь от 2 августа 2010 г.* «Об утверждении положения о порядке определе-ния поставщиков интернет-услуг, уполномоченных оказывать интернет-услуги государственным органам и организациям, использующим в своей деятельности сведения, составляющие государственные секреты»; *При-каз Оперативно-аналитического Центра при Президенте Республики Бела-русь* от 27 мая 2013 г. № 33 «Об утверждении Инструкции о порядке взаимо-действия ведомственных систем электронного документооборота с системой межведомственного электронного документооборота государственных орга-нов», *Приказ Оперативно-аналитического Центра при Президенте Респуб-лики Беларусь* от 30 августа 2013 г. № 62 «О некоторых вопросах техниче-ской и криптографической защиты информации», *Приказ Оперативно-*

*аналитического Центра при Президенте Республики Беларусь* от 10 декабря2015 г. № 11 «Об утверждении Положения о Государственной системе управления открытыми ключами проверки электронной цифровой подписи Республики Беларусь»;

ж) нормативные правовые акты субъектов, органов местного само-управления и т. д.

*Правовой элемент* системы организации защиты информации напредприятии основывается на нормах информационного права и предполага-ет юридическое закрепление взаимоотношений фирмы и государства по по-воду правомерности использования системы защиты информации, фирмы и персонала по поводу обязанности персонала соблюдать установленные меры защитного характера, ответственности персонала за нарушение порядка за-щиты информации.

*Правовая защит*а включает:

* + наличие в организационных документах фирмы, правилах внутрен-него трудового распорядка, трудовых договорах, в должностных инструкциях положений и обязательств по защите конфиденциальной информации;
  + формулирование и доведение до сведения всех сотрудников поло-жения о правовой ответственности за разглашение конфиденциальной инфор-мации, несанкционированное уничтожение или фальсификацию документов;
  + разъяснение лицам, принимаемым на работу, положения о добро-вольности 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*, ориенти-рованные на обеспечение информационной безопасности:

20

* Политика Информационной безопасности;
* Положение о коммерческой тайне;
* Положение о защите персональных данных;
* Перечень сведений, составляющих конфиденциальную информа-

цию;

* Инструкция о порядке допуска сотрудников к сведениям, состав-ляющим конфиденциальную информацию;
* Положение о специальном делопроизводстве и документообороте;
* Обязательство сотрудника о сохранении конфиденциальной ин-

формации;

* Памятка сотруднику о сохранении коммерческой тайны.

1. Международные стандарты, например:

а) [BS 7799-1:2005](http://ru.wikipedia.org/wiki/BS_7799-1) – Британский стандарт BS 7799 Part 1 – *Code of* *Practice for Information Security Management* (Практические правила управле-ния информационной безопасностью) описывает 127 механизмов контроля, необходимых для построения системы управления информационной безо-пасностью организации, определенных на основе лучших примеров мирового опыта в данной области. Этот документ служит практическим руководством по созданию СУИБ;

б) [BS 7799-2:2005](http://ru.wikipedia.org/w/index.php?title=BS_7799-2&action=edit&redlink=1) – Британский стандарт BS 7799 Part 2 – *Informa-tion Security management* –specification for information security managementsystems (Спецификация системы управления информационной безопасно-стью) определяет спецификацию СУИБ. Вторая часть стандарта использует-ся в качестве критериев при проведении официальной процедуры сертифика-ции СУИБ организации;

в) [ISO/IEC 17799:2005](http://ru.wikipedia.org/wiki/ISO/IEC_17799) – «Информационные технологии – Техноло-гии безопасности – Практические правила менеджмента информационной безопасности». Международный стандарт, базирующийся на BS 7799-1:2005; г) [ISO/IEC 27001:2005](http://ru.wikipedia.org/w/index.php?title=ISO/IEC_27001:2005&action=edit&redlink=1) – «Информационные технологии – Методы обеспечения безопасности – Системы управления информационной безопас-ностью – Требования». Международный стандарт, базирующийся на BS

7799-2:2005;

д) [ISO/IEC 27002](http://ru.wikipedia.org/wiki/ISO/IEC_27002) – Сейчас: ISO/IEC 17799:2005. «Информационные технологии – Технологии безопасности – Практические правила менеджмен-та информационной безопасности». Дата выхода – 2007 год;

е) [ISO/IEC 27005](http://ru.wikipedia.org/w/index.php?title=ISO/IEC_27005&action=edit&redlink=1) – Сейчас: BS 7799-3:2006 – Руководство по ме-неджменту рисков ИБ;

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

**Организационно-технические и режимные меры и методы.** Дляпостроения Политики ИБ рассматривают следующие направления защиты ИС:

21

* 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.), информаци-онно-аналитическую деятельность по выявлению внутренних и внешних уг-роз.

Рассмотрим основные особенности разработки политики безопасно-сти учреждения на примере банка7.

На первом этапе производится *изучение фактического состояния ор-ганизации по обеспечению информационной безопасности*.Здесь целью яв-

ляется определения уязвимых мест, угроз информационной безопасности банка по следующим основным направлениям:

 организационная структура банка и нормативно-распорядительная документация в области безопасности, их соответствие це-лям информационной безопасности;

 система управления доступом, в том числе угрозы несанкциони-рованного 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.);

 безопасность компьютерных систем, в том числе проведение ис-следований специальными средствами сканирования безопасности в сети на предмет:

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

oугроз несанкционированного доступа к информации;

oугроз утечки информации при передаче по каналам связи;

oугроз нарушения целостности информации;

oугроз нарушения устойчивой работы информационной системы в

целом;

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

o возможности проведения атак со стороны Интернет посторонни-ми пользователями (злоумышленниками) и др.

1. http://www.cbt.by/main.aspx?guid=1841 (доступ 20.05.2014)

22

Ущерб от НСД (атаки) может быть представлен положительным чис-лом в приблизительном соответствии с таблицей 2.2.

Таблица 2.2 Условная численная шкала для оценки ущерба банку от

НСД

|  |  |  |
| --- | --- | --- |
| Величина | Описание |  |
| ущерба |  |
|  |  |
|  |  |  |
| 0 | Раскрытие информации принесет ничтожный моральный и |  |
| финансовый ущерб банку (фирме) |  |
|  |  |
|  |  |  |
| 1 | Ущерб от атаки есть, но он незначителен, основные фи- |  |
| нансовые операции и положение банка на рынке не затронуты |  |
|  |  |
|  |  |  |
|  | Финансовые операции не ведутся в течение некоторого |  |
| 2 | 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 ухо- |  |
| дит ощутимая часть клиентов |  |
|  |  |
|  |  |  |
|  | Потери очень значительны, банк на период до года теряет |  |
| 4 | положение на рынке. Для восстановления положения требуются |  |
|  | крупные финансовые займы |  |
|  |  |  |
| 5 | Банк прекращает существование |  |
|  |  |  |

Вероятность НСД (атаки) представляется в приблизительном соответ-ствии с таблицей 2.3.

Таблица 2.3. Вероятностно-временная шкала реализации несанкцио-нированного доступа к информационным ресурсам

|  |  |  |  |
| --- | --- | --- | --- |
| Вероятность | Средняя частота события (НСД) | |  |
| события |  |
|  |  |  |
|  |  | |  |
| 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. Ее примерный вид показан в таблице

2.4.

23

Таблица 2.4. Таблица рисков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Описание атаки | Ущерб | Вероятность | Риск |  |
| (Ущерб\*Вероятность) |  |
|  |  |  |  |
|  |  |  |  |  |
| Спам (переполнение | 1 | 0.4 | 0.4 |  |
| почтового ящика) |  |
|  |  |  |  |
|  |  |  |  |  |
| Копирование жестко- |  |  |  |  |
| го диска из централь- | 3 | 0.1 | 0.3 |  |
| ного офиса |  |  |  |  |
|  |  |  |  |  |
| … | … | … | … |  |
|  |  |  |  |  |
| Итого : |  |  | 0.9 |  |
|  |  |  |  |  |

На этапе анализа таблицы рисков (табл. 2.4) задаются некоторым мак-симально допустимым риском (порог), например, значением 0.5.

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

Если интегральный риск (итого) превышает допустимый уровень, значит в системе безопасности набирается множество мелких проблем, кото-рые 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 разрабатывается *концепция информационной безо-пасности* (банка,фирмы,предприятия)и направления ее практической реа-лизации.

Концепция является базовым нормативно-информационным докумен-том и служит основой для:

* создания единой системы правовых, организационных, техниче-ских, режимных и иных мер, обеспечивающих защищенность организации в информационной сфере;
* разработки программ и мероприятий по обеспечению информа-ционной безопасности различных объектов, подготовки локальных норма-тивных документов и политик.

Объектами защиты являются все элементы информационно-технологической инфраструктуры организации (банка), а именно: помеще-ния, 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, на-рушение и несанкционированный доступ к которым ведут к ущербу и поте-рям бизнеса.

24

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

Таким образом, ***в каждом конкретном случае информационная*** ***безопасность должна рассматриваться системно, как комплекс инст-рументов 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 систе-ме.
3. Доступ к внутренним ресурсам информационной системы долж-ны иметь только авторизованные субъекты согласно своим правам.

Направления практической реализации политики безопасности банка можно представить следующим образом8.

* защита помещений и объектов от несанкционированного досту-

па;

* защита информации в бумажном документообороте банка;
* защита информации от утечки по техническим каналам, возмож-ные угрозы при проведении мероприятий и эксплуатации технических средств, когда возможны утечки или нарушения целостности информации, нарушения работоспособности технических средств:

o побочные электромагнитные излучения информативного сигнала от технических средств;

o электрические сигналы или радиоизлучения, обусловленные воз-действием на автоматизированную систему (АС) банка высокочастотных сигналов, по эфиру и проводам, либо сигналов промышленных радиотехни-ческих устройств (радиовещательные, радиолокационные станции, средства радиосвязи и т.п.), и модуляцией их информативным сигналом (облучение, «навязывание»);

o радиоизлучения или электрические сигналы от внедренных спе-циальных электронных устройств перехвата информации;

oакустическое излучение информативного речевого сигнала;

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

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

1. http://www.cbt.by/main.aspx?guid=1841 (доступ 20.05.2014)

25

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

o непреднамеренное прослушивание без использования техниче-ских средств разговоров, ведущихся в выделенном помещении, из-за недос-таточной звукоизоляции его ограждающих конструкций, систем вентиляции

* кондиционирования воздуха;
* обеспечение безопасности информации, передаваемой по кана-лам связи и в системе внутреннего электронного документооборота между подразделениями банка;
* обеспечение информационной безопасности в компьютерных се-

тях;

* обеспечение информационной безопасности в платежных систе-

мах;

* обеспечение информационной безопасности в платежных систе-мах с использованием банковских пластиковых карточек:

o меры технологического характера, имеющие целью повысить на-дежность процедуры передачи информации платежной системы и идентифи-кации владельца карточки,

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

o меры безопасности при обслуживании терминального оборудо-вания (банкоматы, инфокиоски, терминалы), включая меры по предупрежде-нию и/или обнаружению противоправных действий с терминальным обору-дованием, совершаемых третьими лицами в процессе его функционирования,

o организация распределения и закрепления за персоналом рекви-зитов доступа (ключей, служебных карточек операторов и администраторов, паролей, прав доступа);

* обеспечение информационной безопасности при использовании ресурсов Internet:

oиспользование сети Internet сотрудниками банка,

o доступ к сервисам банка клиентов через Internet (Internet-banking),

o использование электронной почты, как внутрикорпоративной, так и через Internet;

* обеспечение безопасности информации в автоматизированных

системах;

* архивы электронной информации;
* размещение компьютерного и телекоммуникационного оборудо-

вания;

* регистрация пользователей и использование паролей на уровне операционных систем, прикладного программного обеспечения и компью-терной сети;

26

* управление доступом на уровне операционных систем, приклад-ного программного обеспечения и компьютерной сети;
* средства предотвращения НСД для операционных систем, при-кладного программного обеспечения и компьютерной сети;
* защита от вирусов и других вредоносных программ;
* системы мониторинга безопасности операционных систем, при-кладного программного обеспечения и компьютерной сети;
* системы сканирования безопасности операционных систем, при-кладного программного обеспечения и компьютерной сети;
* средства аудита безопасности операционных систем, прикладно-го программного обеспечения и компьютерной сети;
* анализаторы протоколов, ведущихся операционными системами, прикладным программным обеспечением и устройствами компьютерной се-ти;
* порядок приемки-сдачи общесистемного и прикладного про-граммного обеспечения;
* установка обновлений на общесистемное и прикладное про-граммное обеспечение;
* резервное копирование, хранение и уничтожение носителей ин-

формации;

* обеспечение непрерывной работы и восстановления программно-технических комплексов;
* рекомендации о службах информационной безопасности и их со-ставу на всех уровнях (головной банк, филиал, отделение).

**Аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ**.В определенной степени этого аспекта проблемымы 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 сетей:

а) системы мониторинга сетей;

б) анализаторы протоколов;

1. антивирусные средства:

а) антивирусные программы;

б) программные и иные антиспамовые средства; в) межсетевые экраны;

1. криптографические средства: а) шифрование данных; б) электронная цифровая подпись;
2. системы бесперебойного питания:
3. системы аутентификации:

27

а) пароль;

б) ключ доступа (физический или электронный);

в) биометрия (анализаторы отпечатков пальцев, анализаторы сет-чатки глаза, анализаторы голоса, анализаторы геометрии ладони и др.).

*Вопросы для контроля и самоконтроля*

1. К чему приводят электромагнитные или ионизирующие излуче-

ния?

1. Привести пример источников электромагнитных или ионизи-рующих излучений.
2. Что такое «преднамеренная помеха», «непреднамеренная поме-

ха»?

1. Из каких основных частей состоит периметр современной ИС?
2. Как можно осуществить неавторизованный доступ к сетевому

трафику?

1. К каким последствиям могут привести отказы системы электро-снабжения ИС?
2. Что такое «компьютерный вирус»? Чем он отличается от осталь-ных деструктивных программ? Привести примеры известных вирусов.
3. Охарактеризовать известные Вам деструктивные программные

средства.

1. Охарактеризовать и привести примеры критических информаци-онных систем.
2. В чем сущность системного подхода при проектировании поли-тики безопасности?
3. Дать классификацию методов и средств защиты информации.
4. Перечислить основные направления реализации Политики ин-формационной безопасности.
5. Перечислить и охарактеризовать концептуальные аспекты инфор-мационной безопасности Республики Беларусь.
6. В чем назначение организационных методов защиты информа-

ции?

1. В чем назначение правовых методов защиты информации?
2. Состав правовой защиты информации.
3. Особенности международного классификатора компьютерных преступлений.
4. В чем назначение режимных методов защиты информации?
5. Какие национальные правовые акты, регламентирующие доступ к информационным ресурсам, Вы знаете?
6. Какие международные правовые акты, регламентирующие дос-туп к информационным ресурсам, Вы знаете?
7. В чем заключается назначение организационно-технических ме-тодов защиты информации?
8. Как можно защитить каналы связи?

28

1. Как можно защититься от мешающих электромагнитных излуче-

ний?

1. Перечислить известные методы и средства авторизации.
2. Назначение межсетевых экранов.
3. Назначение источников бесперебойного питания.
4. На чем основаны биометрические средства идентификации?

29

**ГЛАВА II**

**КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ**

**ЗАЩИТЫ ИНФОРМАЦИИ**

1. **Сущность криптографического преобразования информации**

***3.1. Основы понятия предметной области. Цели и задачи криптографии***

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *X*k |  | *Y*k | ПС |  |
| ИcС | Канал |  |
|  |  |  |
|  |  | передачи |  |  |  |
|  |  |  |  |  |  |



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

Под определение ИС (ИВС) подпадает любая система обработки ин-формации. Далее будем рассматривать ИС как совокупность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.

Математической основой описания и анализа процессов в ИС в широ-ком смысле является теории информации. Возникновение теории информа-ции связывают обычно с появлением фундаментальных работ К. Шеннона9.

**Определение 3.1**. **Построение сигнала по определенным правилам, обеспечивающим соответствие между сообщением и сигналом, несущим это сообщение, называют *кодированием*.**

Кодирование в широком смысле – *преобразование сообщения в сиг-*

*нал*.

Кодирование в узком смысле – *представление исходных знаков*, назы-ваемых символами, в другом алфавите.

1. См., например, *Шеннон К.* Работы по теории информации и кибернетике. — М.: Изд-во иностранной лите-

ратуры, 1963. — 830 с.

30

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* (Xk)до его передачи по каналусвязи:

* кодирование или помехоустойчивое кодирование,
* сжатие или архивирование сообщений,
* криптографическое преобразование10.

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

* + течение последних 20 лет наблюдается бурное развитие открытых академических исследований в этой области. Пока обычные граждане ис-пользовали «классическую» криптографию, «компьютерная» криптография, еще со времен Первой мировой войны, применялась исключительно в воен-ных целях.

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

* в отдельное направление математики, явилась публикация в уже упоми-навшейся работы К. Шеннона «Теория связи в секретных системах».

Современная криптография широко используется и за стенами воен-ных ведомств. Рядовые пользователи информационных технологий получили возможность защититься от всемогущих вредителей (не только от хакеров или кракеров). Ниже проанализируем основные особенности и оценим эф-фективность криптографических методов преобразования (защиты) инфор-мации.

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

Криптографические методы нашли широкое применение в практиче-ской информатике для решения многочисленных проблем *информационной* *безопасности*.В проблематике современной криптографии можно выделитьследующие три типа основных задач:

1) обеспечение *конфиденциальности* (*секретности*),

2) обеспечение *анонимности* (*неотслеживаемости*),

3) обеспечение *аутентификации* информации и источника сообще-

ния.

Первый тип задач относится к защите информации от *несанкциониро-ванного доступа* по секретному *ключу*.Доступ к информации(информаци-

1. Сущность и примеры реализации методов, относящихся к первым двум из перечисленных классов преоб-разования информации, можно найти, например, в [8,9]

31

онным ресурсам) имеют только обладатели ключа. Второй и третий типы за-дач обязаны своей постановкой массовому применению электронных спосо-бов обработки и передачи информации (банковская сфера, электронная ком-мерция, каналы межличностной коммуникации и др.).

Криптографическое преобразование, как и два иных из числа выше-упомянутых (помехоустойчивое кодирование и сжатие), состоит из двух эта-пов: прямого и обратного. Прямое преобразование называют *шифрованием* или *зашифрованием* (в соответствии со стандартом ISO 7492-2 – *зашифрование*, encrypt),обратное- *расшифрованием* или *дешифрованием* (*расшифрование*, decrypt). Процесс передачи зашифрованных сообщений ил-люстрирует рис. 3.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *M* |  | *C* | Расшифрование | *M* |  |
| Зашифрование |  |
|  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



*K*1 *K*2

Рис.3.1. Общая структурная схема криптосистемы

Исходное сообщение называется открытым текстом (*М*, от английско-го message). Зашифрованное сообщение – *шифртекстом* или *шифрограммой*

* (от английского cipher). После обратного преобразования получаем исход-ный (или приближенный к нему) документ (*М’*). Таким образом, в канал пе-редается шифртекст, *С*.

Если пользоваться символьными обозначениями, введенными нами ранее (*Хk* – данные до прямого преобразования, *Хn* – данные после прямого

преобразования), т.е. положить *X* *k*  *M* и *X* *n*  *C* *,* то сравнительной харак-

теристикой криптографических методов будет равенство *k*  *n* . Это означает, *что длина открытого и зашифрованного сообщений не меняется* (исключе-ние составляют методы шифрования с использованием электронной цифро-вой подписи).

Функция зашифрования, *Е*, в математическом виде представляется следующим образом:

|  |  |
| --- | --- |
| *Е(М) = С*, | (3.1) |
| В обратном процессе функция расшифрования, *D*, восстанавливает *М*: | |
| *D(C) = M.* | (3.2) |

Поскольку смысл зашифрования и последующего расшифрования со-общения заключается в восстановлении исходного открытого текста, спра-ведливо следующее равенство:

*D(Е(М)) = M.*

* этом анализе прямого и обратного процессов преобразования шифр отождествляется с *криптографическим алгоритмом*, представляющим собой

32

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 мес-тами. Первые алгоритмы относятся к классу *подстановочных*, другие – *перестановочных*.Современные криптосистемы используют как подстанов-ки, так и перестановки символов.

***3.2 Подстановочные и перестановочные шифры***

***3.2.1 Подстановочные шифры***

Сущность подстановочного шифрования состоит в том, что, как пра-вило, исходный текст (*М*) или зашифрованный текст (*С*) используют один и тот же алфавит, а тайной является алгоритм подстановки. Такой шифр назы-вается 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» меня-ется 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 расшифрова-ния необходимо выполнить обратную замену. Как видим, такая криптоси-стема 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 (mod N)*, | (3.3) |
| *x = y + k (mod N)*, | (3.4) |

где *х* и *y* — соответственно порядковые номера (индексы) символов открыто-го и зашифрованного текстов, *N* — мощность алфавита (кол-во символов), *k*

— ключ.

*Пример 3-1*.Имеем открытый текст *М*=«*cba*».На основе шифра Цеза-ря *С* = «*fed*».

Здесь *k =*3, *N =* 26. Первый символ открытого текста (*с*) имеет индекс

1. (помним, что начальный символ алфавита (*а*) имеет нулевой индекс). Зна-чит, первый символ шифртекста (*с*) будет иметь индекс 2 + *k* = 5. А такой ин-декс 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 зашифро-ванном виде так:

33

*YHQL YLGL YLFL*.

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

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

Как видим, в шифре Цезаря использовались только аддитивные свой-ства множества целых чисел. Однако концепция, заложенная в систему шиф-рования 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 подстано-вок Цезаря.*** Определим преобразование в такой системе:

*y = a\*x + b (mod N)*,

где *a* и *b* – целые числа.

При этом взаимно однозначные соответствия между открытым тек-стом и шифртекстом будут иметь место только при выполнении следующих

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| условий: | | | 0 ≤ *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, *а* = 3, *b* =5.Тогда,НОД(3,26)=1,и мы по- | | | | | | | | | | | | | | | | | | | | | | |
|  | |  | | | |  | | | | | | | | | | | | |  |  |  |  |  |  |
| лучаем следующее соответствие между индексами букв: | | | | | | | | | | | | | | | | | | |  |  |  |  |  |  |
| х | **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** |
| 3х+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 следую-щее соответствие для букв открытого текста и шифртекста:

**А** **B C** **D**

**Е**

**F Q** **Н**

**I**

**J** **К**

**L** **М** **N** **0**

**Р**

**Q R** **S**

**T**

**U V W**

**Х**

**Y Z**

**F** **I**

**L O R U** **Х А** **D** **Q**

**J**

**М Р**

**S V Y** **В Е Н** **K**

**N Q**

**Т**

**W Z**

**C**

* соответствии с последней таблицей открытому тексту *BELSTU* бу-дет соответствовать зашифрованный: *IRMHKN*.

***Система шифрования Цезаря с ключевым словом*** является одноал-

фавитной системой подстановки. Особенностью этой системы является ис-пользование *ключевого слова* для смещения и изменения порядка символов в алфавите подстановки (желательно, чтобы все буквы ключевого слова были различными) и некоторого числа *а*: 0 ≤ *a < N*. Рассмотрим систему на приме-ре.

34

*Пример 3-3*.Выберем некоторое число *а*,и слово или короткую фразу

* качестве *ключевого слова*. Пусть выбраны слово *DIPLOMAT* в качестве ключевого и число *а* = 5.

Ключевое слово записывается под буквами алфавита, начиная с бук-вы, индекс которой совпадает с выбранным числом *а*, как это показано в ни-жеследующей таблице:

**0 1 2 3 4 5**

**А** **B C D** **Е** **F Q** **Н**

**I**

**10 15 20 25 J К L М N 0 Р Q R S T U V W Х Y Z**

**D I P**

**L O M A** **T**

Оставшиеся буквы алфавита подстановки записываются после ключе-вого слова в алфавитном порядке:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **5** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **А** | | | **B C** | | | | **D** | | | **Е** | | | **F** | **Q Н** | **I J К L М N** | |  | **0 Р** | | | **Q R** | | | |  | **S** | **T** | | | **U** |  | **V** | **W Х** | | | |  | **Y** |  | **Z** |
|  | | |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **V** |  | **W** |  | **Х** |  |  | **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 правило, которое мы рассматривали на выше проанализированных примерах.

Возникает вопрос: а что будет, если примять несколько ключей при зашифровании/расшифровании одного сообщения? Тогда будет система из-вестная как *шифр Виженера11*.

***Шифр Виженера***.В этом шифре,как было отмечено,мы имеем дело

* последовательностью сдвигов, циклически повторяющейся. Основная идея заключается в следующем. Создается таблица (*таблица Виженера*) размером *N\*N* (*N* –число знаков в используемом алфавите).Эти знаки могут включатьне 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 сим-вол влево. Таким образом, при мощности алфавита (английского языка) рав-ной 26, необходимо выполнить последовательно 25 сдвигов для формирова-ния всей таблицы.

Вид такой таблицы (*квадрата Виженера*) представлен на рисунке 3.2. 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 соответствующего ему (с тем же индексом) символа ключе-

1. французский дипломат Блез Виженер (фр. *de Vigenère)* представил описание простого, но стойкого шифра перед комиссией Генриха III во Франции в 1586 году

35

вой последовательности (выбирается из верхней строки таблицы). Например, если первым символом сообщения будет символ *В*, а первым символом клю-ча будет символ Т, то первым символом шифртекста будет символ *U* (нахо-дится на пересечении 2-й строки – *В* – и 20-го столбца – *Т* таблицы): здесь *k* = 18; при зашифровании символа *Е* выбирается 5 строка и, если вторым симво-лом ключа будет *I* (9-ый столбец матрицы), то вторым символом шифртекста будет *М*: здесь уже *k* = 4. При расшифровании следует использовать (3.4).

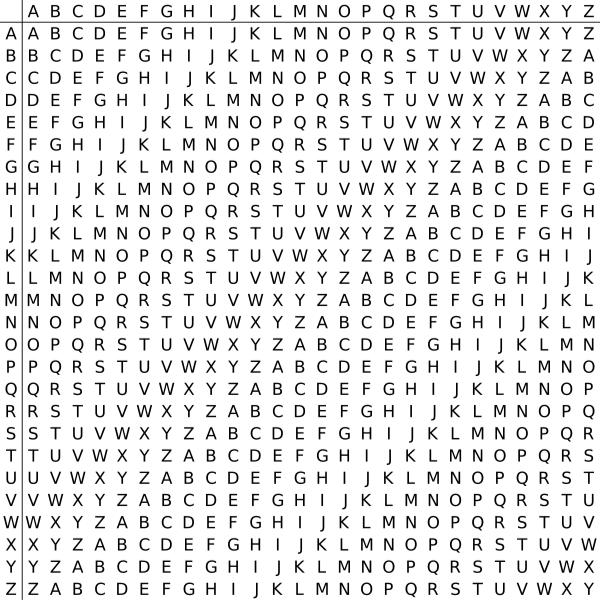


Рисунок 3.2 Таблица (квадрат) Виженера

Именно такая таблица, а не одна строка, как в шифре Цезаря, исполь-зуется для зашифрования/расшифрования сообщения. Таким образом, в оп-ределенном 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 зашифрова-ния/расшифрования также проанализируем с помощью примера.

*Пример 3-4*.Предположим,что шифруемое сообщение имеет вид:М=«*BELSTU*». Как и в *системе шифрования Цезаря с ключевым словом,* в на-шем случае необходимо выбрать ключ. Пусть ключом будет слово «*TIR*». Желательно выбирать ключевое выражение, имеющее длину сообщения с минимальными повторениями одинаковых символов, что влияет на крипто-стойкость шифра при вероятностном криптоанализе. Если же, как в нашем случае, ключ имеет меньшую длину (3 символа в сравнении с 6), то его сле-

36

дует 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** |

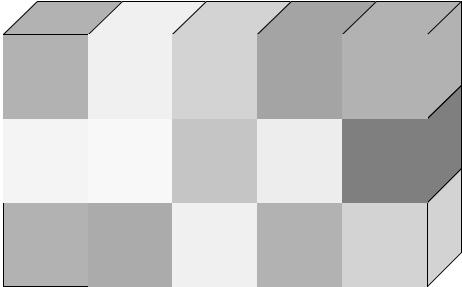
Даже из этого простого примера видно, что одинаковым буквам со-общения (*S, U*) соответствует одна и та же буква в шифртексте (*L*).

Следует отметить, что для выполнения эффективного частотного (ве-роятностного) криптоанализа аналитику важно знать длину ключа, если его начальная длина (как в последнем примере) меньше длины шифруемого со-общения.

***3.2.2 Перестановочные шифры***

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

*Пример 3-5*.М=«ВАСЯ ЛЮБИТ МАШУ».Запишем этот текст как по-казано на рис. 3.3.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| В | А | С | Я | \_ |
| Л | Ю | Б | И | Т |
| \_ | М | А | Ш | У |

Рис. 3.3. Использование простого вертикального перестановочного шифра

Считывание по столбцам снизу вверх приводит к такому шифртексту:

С=«\_ЛВМЮААБСШИЯУТ\_».

Принцип записи исходного сообщения и порядок считывания симво-лов может быть различным. Обратимся к следующему примеру.

*Пример 3-5*.Открытый текст возьмем из предыдущего примера,а за-пишем его так, как показано на рис. 3.4.

37

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | С |  | Ю |  | \_ | У |  |
|  |  |  |  |  |
| А | Я | Л | Б | Т | М | Ш |  |
| В |  | \_ |  | И |  | А |  |
|  |  |  |  |  |

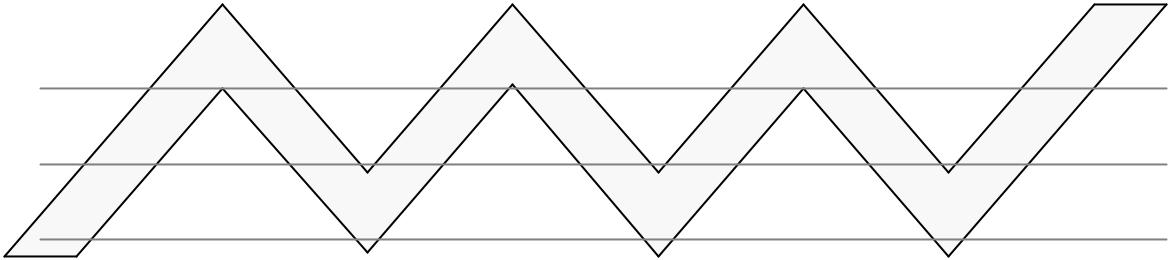


Рис. 3.4. Использование перестановочного шифра

Осуществляя считывания по уровням, начиная с верхнего, получим следующий шифртекст: *С*=«СЮ\_УАЯЛБТМШВ\_ИА».

Существуют еще более сложные перестановочные шифры, но компь-ютеры достаточно быстро справляются с ними. При этом использование дан-ных шифров требует большого объема памяти.

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

Ограниченные алгоритмы не допускают эффективного контроля пли стандартизации. Каждая группа пользователей должна использовать собст-венный уникальный алгоритм. Такие группы не могут использовать откры-тые аппаратные или программные продукты - злоумышленник может приоб-рести такой же продукт и раскрыть алгоритм. Этим группам приходится раз-рабатывать и реализовывать собственные алгоритмы.

Несмотря на указанные фундаментальные недостатки, ограниченные алгоритмы необычайно популярны в приложениях с низким уровнем заши-ты. 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, ко-торый на рис.3.1 обозначен буквой *К*. Такой ключ может быть любым значе-нием, 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, *Е*: |  |
| *ЕК(М) = С*, | (3.5) |
| функция расшифрования, *D*: |  |
| *DК(C) = M* | (3.6) |

или

38

*DК(ЕК(М)) = M.*

Соотношения (3.5)-(3.6) описывают криптосистемы, использующие одинаковое значение ключа для зашифрования и расшифрования (*К*1 = *К*2 = *К*).Такие криптосистемы называются *симметричными*.

Простейшим примером симметричного криптопреобразования явля-ется сложение по модулю два сообщения *М* с ключом *К* для получения шиф-ртекста. Расшифрование достигается выполнением такой же операции над *С*

* *К*.

**Симметричные криптосистемы называются также криптосисте-мами с *тайным ключом,* поскольку значение ключа должно быть из-вестно только отправителю и получателю сообщений*.***

Другой класс современных криптоалгоритмов для зашифрования и расшифрования использует различные ключи (*К*1 ≠ *К*2 ).

* этом случае формальное представление анализирумых преобразо-ваний представляется следующим образом:

*функция зашифрования*, *Е*:

|  |  |
| --- | --- |
| *ЕК1(М) = С*, | (3.7) |
| *функция расшифрования*, *D*: |  |
| *DК2(C) = M* | (3.8) |
| или |  |
| *DК2(ЕК1(М)) = M.* | (3.9) |

**Ключи *К*1 и *К*2 являются разными, но взаимозависимыми (один из них тайной не является). Поэтому асимметричные криптосистемы на-зывают 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). Та-кие алгоритмы называют *потоковыми*. Другие алгоритмы обрабатывают группы (блоки) битов открытого текста. Эти алгоритмы называют *блочными*.

* 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 от-крытого 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 несанкциониро-ванный доступ к информации (*атаку* на систему).

39

**Задачей криптоанализа является восстановление открытого тек-ста без доступа к ключу. Попытка криптоанализа также называется *атакой*.А раскрытие ключа без привлечения специальных методов на-зывают *компрометацией ключа*.**

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

Голландский криптограф Керкгоффс12 (1835-1903) впервые сформу-лировал постулаты оценки *стойкости шифра* (или *криптостойкости*) перед его *взломом* (раскрытием), в соответствии с которыми:

1. весь механизм преобразования считается известным злоумышлен-

нику (*интрузу*),

1. *криптостойкость* (надежность)алгоритмов преобразования опре-деляется 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.

* 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 располага-

ет шифртекстами нескольких сообщений и открытыми текстами этих же со-общений. Его задача – определить ключ.

*Атака на основе выбранного открытого текста* предполагает воз-

можность доступа криптоаналитика не только к шифртекстами нескольких сообщений и открытым текстам этих же сообщений, но и возможность выби-рать открытый текст для зашифрования. Задача – определить ключ.

*Атака на основе выбранного шифртекста* позволяет криптоаналити-

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

* случае *атаки на основе подобранного ключа* криптоаналитик кое-

что знает о связях между ключами.

Сложность той или иной атаки можно оценит по следующим критери-

ям:

*по сложности данных* –оценивается объем данных,необходимых дляреализации атаки,

1. *Auguste Kerckhoffs.* La cryptographie militaire.—1883

40

*по сложности обработки* –оценивается время,необходимое на реа-лизацию атаки (фактор трудозатрат),

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

Ларс Кнудсен (Lars Knudsen) классифицировал сложность взлома ал-горитмов по нескольким критериям:

*полное вскрытие* –криптоаналитик находит ключ,такой что *DК(C) = M* (см. (3.6));

*глобальная дедукция* –криптоаналитик находит альтернативный алго-ритм, эквивалентный (3.6) без знания *К*;

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

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

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

Таким алгоритмом реализован в *шифре Вернама* или *шифре на основе* *одноразовых блокнотов*.Такой шифр предложили в1917году Гилберт Вер-нам (Gilbert Verman) и Мэйджор Моборн (Major Mauborgne). Классический одноразовый блокнот – это не повторяющийся 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 ос-мысленности полученного открытого текста. Такой метод называется *лобо-вой атакой* (от англ. brute-force).Время,необходимое для любового вскры-тия системы, зависит от двух параметров: числа тестируемых ключей и вре-мени тестирования отдельной ключевой комбинации.

*Вопросы для контроля и самоконтроля*

1. Охарактеризовать обобщенную структурную схему информаци-онной системы. В чем состоит ее отличие от ИС с криптографическим пре-образованием информации?
2. Сформулировать основную задачу криптографии.
3. Сформулировать основную задачу криптографии.
4. Охарактеризовать основные элементы криптосистемы.
5. Дать классификации шифров по различным признакам.
6. Создать собственный подстановочный шифр и оценить его крип-тостойкость.
7. Создать таблицу Виженера для алфавита русского (белорусского)

41

языка. По указанию преподавателя (или самостоятельно) выбрать ключ и за-шифровать сообщение, состоящее из собственных фамилии, имени и отчест-ва.

1. Создать таблицы, как в предыдущем задании, дополнив алфавит знаком пробела. Выполнить задание, сформулированное в п.7.
2. Создать собственный перестановочный шифр и оценить его криптостойкость.
3. На основе каких данных строятся атаки на засекречивающие сис-

темы?

1. Чем определяется сложность взлома криптоалгоритмов?

42

1. **Основы криптографии**

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

***4.1 Элементы теории чисел***

Теория чисел, или высшая арифметика изучает свойства целых чи-сел. К ц*елым* относятся, например, числа … -4, -3, -2, -1, 0, 1, 2, 3, 4, … . *На-туральные числа* –целые положительные числа:1, 2, 3, 4, ….

Некоторое число *a* является *делителем* другого числа, *b*, если *b = a* *\** *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 чис-ла *b*, то говорим: *a* не делит *b*.

Натуральное число *p* называется *простым*, если *p > 1* и не имеет по-ложительных делителей, отличных от 1 и *p*. Простое число не делится без ос-татка ни на одно другое число. Например, простыми являются числа 2, 3, 5, 73, 2521, 2365347734339 и 2756839 – 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*,кроме1, можно представить как произведение простых множителей:

*N = p1 \* p2 \* p3 \* ... \* pn , n > 1* .

Целое число 37 – простое. Целое число 1554985071 = 3 3 4463 38713 – 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 котороеделятся все данные числа без остатка. Наибольший из делителей называется *наибольшим общим делителем* (НОД).

Один из способов вычисления НОД двух чисел базируется на *алго-*

|  |  |  |
| --- | --- | --- |
| *ритме Евклида*. |  |  |
| Пусть даны два числа – *a* и *b*; *a*0, *b* 0, | | считаем, что *a* > *b*. |
| Находим ряд равенств: |  |  |
| *a = b q*1+ *r*1, | 0 *r*1 < *b*, |  |
| *b = r*1 *q*2+ *r*2, | 0 *r*2 < *r*1, |  |
| *r*1 *= r*2 *q*3+ *r*3, | 0 *r*3 < *r*2, |  |
| *r*2 *= r*3 *q*4+ *r*4, | 0 *r*4 < *r*3, |  |
| · · · · · · · · · · · · | · · · · · ·· | (4.1) |

43

|  |  |
| --- | --- |
| *rn*–3 *= rn*–2 *qn*–1+ *rn*–1, | 0 *rn*–1< *rn*–2 , |
| *rn*–2 *= rn*–1 *qn* + *rn*, | 0 *rn* < *rn*–1, |
| *rn*–1 *= rn qn*+1, | *rn*+1 *=* 0, |

заканчивающийся, когда получаем некоторое *rn*+1 = 0. Тогда *rn* – наиболь-ший общий делитель чисел а и *b*.

Последнее неизбежно, так как ряд *b*, *r*1, *r*2, …, как ряд убывающих це-лых, не может содержать более чем *b* положительных. Имеем: *b* > *r*1 > *r*2 > ...

* *rn* 0,следовательно,процесс оборвется максимум через *b* шагов.

*Пример 4-1****.***Пусть *а* = 525, *b* = 231.Найти НОД.Применим алго-ритм Евклида:

525 = 231 2 + 63;

231 = 63 3 + 42;

63 = 42 1 + 21;

42 = 21 2.

Получаем последний положительный остаток *r*3 = 21. Таким об-разом, НОД (525, 231) = 21.

*Пример 4-2.* Пусть а= 1234, *b* = 54.Найти НОД.

1234 = 54 22 + 46;

54 = 46 1 + 8;

46 = 8 5 + 6;

8 = 6 1 + 2;

6 = 2 3.

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

Схема алгоритма приведена на рис. 4.1.

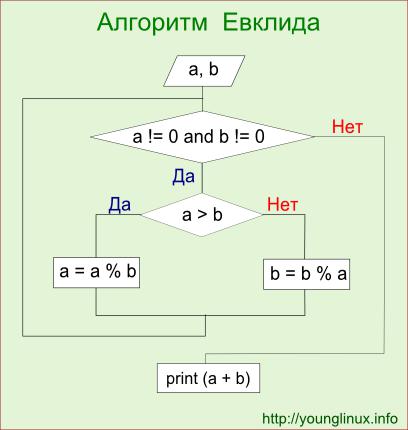


Рисунок 4.1. Блок-схема алгоритма Евклида для вычисления НОД

44

*Взаимно простые числа* —это два или несколько целых чисел,наи-больший 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 чис-лу 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 взаим-но простых с ним можно подсчитать на основе известной *функции Эйлера* (по имени швейцарского математика Леонарда Эйлера, 1707-1783), иногда назы-ваемой «*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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *φ(n) = n-1*. |  | (4.2) |
| Другие примеры: |  |  |  |
| (1) =1; | (5) = 4; | (9) = 6; | |
| (2) = 1; | (6) = 2; | (10) | = 4; |
| (3) = 2; | (7) = 6; | (11) | = 10; |
| (4) = 2; | (8) = 4; | (12) | = 4. |

Любое положительное целое число *р* может быть выражено с по-мощью положительных целых чисел, не превосходящих и взаимно про-

стых с каждым делителем числа *р*. Например, 6 = 2 3 имеет четыре де-лителя: 1, 2, 3 и 6.

(1) + (2) + (3) + (6) = 1 + 1 + 2 + 2 = 6.

Если *n = p\* q*, то

*φ(n) = (p-1)\*(q-1)*. (4.3)

Если числа *р* и *q* – взаимно простые, то

(*p q*) = (*p*) \* (*q*).

Например, пусть *р* = 8 и *q* = 15. Тогда (8) = 4, поскольку

только 1, 3, 5 и 7 – положительные целые числа, которые меньше 8 и взаим-

но простые с 8. Также (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,

(120) = (8) \* (15) = 32,

что можно проверить непосредственно.

* другой стороны, если *p* и *q* *–* очень большие простые числа и извес-тен результат их перемножения (число *n*), то обратная задача – найти *p* и *q* по известному *n* (*задача факторизации*) даже для современных вычислитель-ных средств представляется практически неразрешимой. Эта особенность

45

используется, в частности в некоторых алгоритмах асимметричной крипто-графии.

***4.2 Основы модулярной арифметики***

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

*a ≡ b* (mod *n*) или *a ≡ b* mod *n*.

* + 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 *а* на число *n* справедливо *a = b* + *k\*n.*

*Пример 4-3.* При *a =* 13и *n= 4* имеем *b=*1,т.е. 13 = 1 + 3\*4.Для дан-ного примера справедлив вывод: число 13 по модулю 4 равно 1 или числа 13

* 1 равны по модулю 4.

*Пример 4-4.* При *a =* 13и *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) mod n = ((а mod n )+(b mod n )) mod n, (a – b) mod n = ((а mod n )-(b mod n )) mod n, (a \* b) mod n = ((а mod n )\*(b mod n )) mod n,*

*(a \* (b+с)) mod n = (((а\* b) mod n )+((а\*с) mod n )) mod n.*

Модулярная арифметика, как видим, ограничивает диапазон проме-жуточных 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. Од-нако существуют методы ускорения таких вычислений.

Проиллюстрируем это простым примером.

*Пример 4-5.* Нужно вычислить модуль *n* некоторого числа *а* в8степе-

ни: *а8* *mod n.*

Понятно, что семь операций умножения числа *а* могут дать огромное число. Порядок чисел, которыми оперирует вычислитель, можно значительно

уменьшить, если воспользоваться промежуточными вычислениями по моду-лю: ((*а2* *mod n*)*2* *mod n*)*2* *mod n.*

*Пример 4-6.* Предположим,что показатель степени не является степе-нью 2. Пусть это будет, например, 25, т.е. необходимо вычислить *а25* *mod n.*

46

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

*а25 mod n = (а\* а24) mod n = ((((а2 \* а)2)2)2 а) mod n =*

* + *(((((((а2 mod n ) а) mod n )2 mod n)2 mod n)2 mod n ) а) mod n* .

*Вопросы для контроля и самоконтроля*

1. Привести примеры простых и взаимно простых чисел, двоичное представление которых состоит не мене, чем из 10 бит.
2. Сформулировать основную теорему арифметики.
3. Найти НОД пар чисел: 315 и 196; 294 и 7; 1443 и 143; 2187 и 343.
4. В чем сущность задачи факторизации?
5. Справедливо или не справедливо равенство: (120) = (8) \* (15) = 32? Пояснить ответ.
6. Представить 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 зна-чение является понятием более сложным.

Вспомним, что в анализируемой арифметике запись *a \* х ≡ 1 mod n*

эквивалентна поиску таких значений *х* и *k*, которые удовлетворяли бы тожде-ству:

*a \* х = n\*k +1*.

Общая задача вычисления обратного значения по модулю формирует-ся следующим образом: нужно найти такое значение *х*, которое бы удовле-творяло уравнению

|  |  |
| --- | --- |
| *(a \* х) mod n = 1.* | (4.4) |
| Последнюю формулу можно представить в таком виде: |  |
| *х-1 = a mod n*. | (4.5) |
| Число *х-1* является обратным значением по модулю *n* числа *а*. |  |

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

*Пример 4-7.* Необходимо найти обратное значение числа5по модулю

14.

Перепишем необходимое уравнение в форме (4.4):

*(5 \* х) mod 14 = 1.*

Легко установить, что *х* = 3, т.е. справедливо *3-1* *= 5 mod 14.*

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

**Следует запомнить: 1) уравнения (4.4) и (4.45) имеют единствен-**

**ное решение, если числа *х* 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.***

47

Обратное значение по модулю можно вычислить, воспользовавшись *расширенным алгоритмом Евклида*.Ниже(листинг4.1)приводится реализа-ция этого алгоритма на языке С++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;

 }



\*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;

1. заимствован из книги: Б. Шнайер. Прикладная криптография.- М.:Триумф.-2003.-816 с.

48

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;

 }

Листинг 4.1. Реализация расширенного алгоритма Евклида на языке С++

**Китайская теорема об остатках**. ***В общем случае,* *если разложение* *числа n на простые множители представляет собой p1\*p2\*…\*pt (неко-торые простые числа могут встречаться несколько раз), то система уравнений***

|  |  |
| --- | --- |
| ***(x mod pi) = ai*,где *i* = 1,2…,*t*** | (4.6) |

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

Иными словами число (меньшее, чем произведение нескольких про-стых 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 первом веке н.э. китайским математиком Сун Цзе.

*Пример 4-8.* Положим, *n=*15и *p1=3, p2*=5; *a1* = 2, *a2* = 4.

Запишем систему уравнений:

*(x mod p1) = a1,*

*(x mod p2) = a2;*

или

*(x mod 3) =* 2*,*

*(x mod 5) =* 4*.*

Этот пример показывает, что существует единственное число, мень-шее, чем 3\*5 = 15, с такими вычетами: 14, т.е. два вычета однозначно опреде-ляют число.

**Малая теорема Ферма. Если *n* *–* простое число*,* а число *а* не крат-но *n,* то справедливо**

|  |  |
| --- | --- |
| ***an = 1 mod n***. | (4.7) |

Автором этой теоремы (доказательство опустим) является француз-ский математик Пьер де Ферма (1601-1665).

49

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

|  |  |
| --- | --- |
| *aφ(n)* mod *n = 1*. | (4.8) |
| Последнее выражение можно переписать в следующем виде: |  |
| *а-1* mod *n = aφ(n)-1* mod *n*. | (4.9) |

*Пример 4-7.* Найти число,обратное5по модулю7.

Так как число 7 является простым, то используя (4.2), получим: *φ(7) =*

7 – 1 = 6. Теперь с помощью (4.9) получаем: 56-1 mod 7 = 55mod 7 = 3.

Таким образом, *5-1* mod *7=*3 или *5\*3 = 1 mod 7*.

***4.4 Проблема дискретного логарифма***

Если известны три некоторых числа (*а,* *х*, *n*), то достаточно легко

|  |  |
| --- | --- |
| можно вычислить число *y*: |  |
| *y=ax* mod *n*. | (4.10) |

Обратная задача. Найти *х*, если известны *а,* *у*, *n*. Эта задача решается гораздо труднее. Ее называют *задачей* *(проблемой)* *дискретного логарифми-рования*.По аналогии с вещественными числами,для которых *х* = 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 од-нонаправленных или использования *однонаправленных функций*.

Однонаправленная функция – одно из центральных понятий в асим-метричной криптографии.

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

*Вопросы для контроля и самоконтроля*

1. Сформулировать китайскую теорему об остатках.
2. Представить числа 9, 21, 25, 27, 33, 45, 55, 75 в виде произведе-ния простых чисел и, используя (4.6), найти решение системы уравнений

*(x mod pi) = ai .*

1. Сформулировать и пояснить Малую теорему Ферма.
2. Найти число, обратное *х* по модулю *n*, если *х* и *n* соответственно равны: 4 и 11; 4 и 5; 7 и 3; 19 и 13; 7 и 17.
3. Пояснить сущность проблемы дискретного логарифмирования.

50

1. **Характеристики и реализация криптографических алгоритмов**

***5.1 Алгоритм DES***

* 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 простой пример.

*Пример 5-1.* Пусть открытое сообщение(*М =*{*mi*})в двоичной формеимеет вид: *М* *=*10101100*.* Считаем, что выбран симметричный ключ *К*1*=К*2*=К=*1010.Используется самая простая операция зашифрования:

*ci*  *mi*  *K ,*

и операция расшифрования:

*mi*  *ci*  *K* ,

где *i* *–* *i*-тый блок зашифрования, *mi* – *i*-тая часть сообщения.

Нетрудно убедиться, что *C*  *c*1*c*2  00000110 , а сложение каждых че-тырех бит шифртекста с ключом восстанавливает исходное сообщение (здесь

* - 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 Нацио-нальным 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 бит; дополнитель-но к ним вычисляются 8 бит четности: 8-й бит в каждом из 8 байтов ключа. Такая длина ключа соответствует 1017 комбинаций, что обеспечивало до не-давнего времени достаточный уровень безопасности.

Входной блок данных, состоящий из 64 бит, преобразуется в выход-ной 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 пре-образованию (и при зашифровании, и при расшифровании) в течение 16 ра-ундов. 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 (см. рис. 5.1).

51

* таблице 5.1 показан принцип первоначальной перестановки входно-го 64-битового слова.

Таблица 5.1 Начальная перестановка14

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Выполненная перестановка означает, например, что первый бит вход-ного блока сообщения будет размещен на 40-й позиции, а 58-й – на первой и т.д. Из беглого анализа выполненной перестановки легко понять принцип. Алгоритм перестановки разрабатывался для облегчения загрузки блока вход-ного сообщения в специализированную микросхему.

Поскольку программная реализация рассмотренной перестановки дос-таточно трудна, и с учетом того, что сама процедура не влияет на общую криптостойкость алгоритма, во многих программных приложениях эта пере-становка, как и конечная, из алгоритма исключаются.

Для выполнения упомянутых действий вводится функция *f* (на рис. 5.1

– это *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).

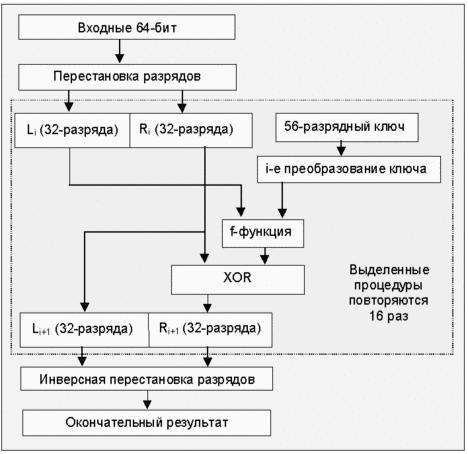


Рисунок 5.1 Общая схема алгоритма DES

1. последовательность выходных бит здесь и в последующих таблицах соответствует числам, читаемым по строкам, затем – по столбцам

52

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

Сначала 32 входные разряда расширяются до 48 (расширяющая пере-становка при этом некоторые разряды повторяются), после чего происходит сложение их по модулю два (ХOR) с ключом (по аналогии с примером 5-1):

|  |  |
| --- | --- |
| *Li=Ri-1,* |  |
| *Ri=Li-1 +f(Ri-1, Ki+1)*. | (5.1) |

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

**Основной криптографический смысл такого преобразования (уд-линения) состоит в том, что увеличивается влияние одного бита сообще-ния на общий результат, т.е. возрастает влияние одного бита сообщения на общий результат. Это явление называется *лавинным эффектом***. **Ла-винный эффект повышает криптостойкость системы.**

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

Результирующий 48-разрядный код преобразуется далее в 32-разрядный с помощью *S*-матриц. На выходе *S*-матриц осуществляется пере-становка 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-битовое слово, значение которого определяется входным 6-битовым словом.

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

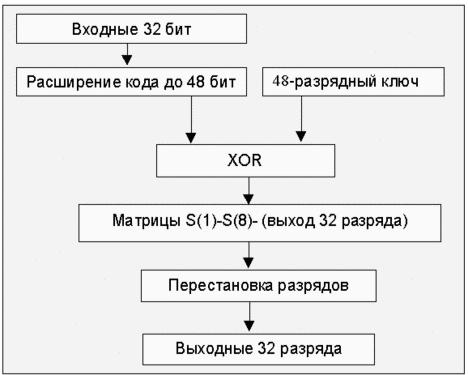


Рис. 5.2. Схема реализации функции *f*

53

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 5.2 Расширяющая перстановка | | | | | | | | |  |  |  |
| 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 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Таблица 5.3. *S*-блоки алгоритма DES | | | | | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | S-БJIOK 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-БJIOK 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-БJIOK 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-БJIOK 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-БJIOK 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-БJIOK 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-БJIOK 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-БJIOK 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 |

54

Входное 6-битовое слово на входе каждого блока задает адрес ячейки соответствующей матрицы, причем 1-й и 6-й биты задают строку, а осталь-

ные – столбец матрицы (адреса начинаются с числа 0). Выходным 4-хби-товым сообщением данного блока будет бинарное представление содержи-мого ячейки таблицы с указанными адресами.

*Пример 5-1*.Определить выходное сообщение8-гоS-блока,если наего вход подается сообщение 101011. Первый и последний биты (11) опреде-ляют третью строку матрицы. Четыре остальных (0101) – адрес столбца. По-следнее двоичное число соответствует десятичному числу 5. Это значит, что следует выбрать 6-й слева (первый имеет нулевой – 0000 – адрес) столбец матрицы. На пресечении указанных строки и столбца находится ячейка, со-держимым которой является число 10 (выделено). Таким образом, выходным значением данного блока будет 4-хразрядное число 1010.

Подстановка с помощью S-блоков является ключевой операцией всего алгоритма. Поскольку она является нелинейной (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 (на рис.5.2 – блок перестановки разрядов или Р-блок). Этот про-цесс перестановки поясняет таблица 5.4.

Таблица 5.4 Перестановка в Р-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) об-ратна по отношению к начальной.

**Преобразования ключей**.Первоначальной64-битовый ключ сокра-щается до 56-битово путем отбрасывания битов четности. Остаются 56 бит, последовательность которых показана в табл. 5.5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Таблица | | 5.5. | Таблица | | начальной | | | перестановки | | | | бит | | 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-битовый подключ. Для этого 56 бит сначала делятся пополам. Затем над каждой поло-виной выполняется операция сдвига на 1 либо 2 бита (в зависимости от раун-да, см. табл. 5.6).

55

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 ра-унда. А в текущем раунде выполняется основная операция – сжимающая пе-рестановка: из 56 бит будет выбраны 48 и их последовательность будет из-менена. Принцип такого преобразования виден из 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 пере-мещается на 35-ю позицию, а 18-й бит отбрасывается.

**Расшифрование в DES**.Осуществляется в обратной последователь-ности. Компоненты алгоритма подобраны так, чтобы для зашифрования и расшифрования применялся одинаковый алгоритм. Понятно, что ключи ис-пользуются в обратном порядке в сравнении с зашифрованием.

**Криптостойкость DES**.Мы выше отмечали,что этот важнейший па-раметр алгоритма определяется не только ключом, но преобразованием блока открытого текста или шифртекста с помощью S-матриц. Что касается ключа

* его преобразований, то рассматриваемому алгоритму свойственна так на-зываемая *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 раз-личных ключей, которые при зашифровании превращают открытые тексты в одинаковый шифртекст.
  + 1976 г. У. Диффи и М. Хеллман утверждали, что специализирован-ный компьютер стоимостью 20 млн долларов сможет восстановить ключ за 1 день.
  + 1990 году Э. Бихам и А. Шамир предложили новый метод криптоа-нализа – *дифференциальный*, который оказался более эффективным в срав-нении с лобовым вскрытием. Криптоаналитик работает с парами шифртек-стов, 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 ра-унды алгоритма с использованием одного и того же ключа.

*Линейный криптоанализ* алгоритма означает следующее.Если выпол-нить операцию XOR над некоторой частью открытого текста, затем над неко-торой частью шифртекста, а затем над полученными результатами (похоже на пример 5-1), то получается некая последовательность, которая представ-

56

ляет собой XOR некоторых битов ключа. Используя фрагменты открытых текстов и соответствующих им шифртектов, можно найти ключ.

Реальность такова, что к началу 90-х годов рассматриваемый алго-ритм был отнесен к числу обеспечивающих низкую криптостойкость.

* настоящее время он все еще находит ограниченное применение.
* приложении 1 приведен программный код алгоритма.15

***5.2 Модификации алгоритма DES***

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

Наиболее простой метод объединения или наращивания DES состоит

* использовании *двух алгоритмов с различными ключами*:

зашифрование: *С* *= EK2(EK1(M)),*

расшифрование: *M = DK1(DK2(M)).*

Криптостойкость такого преобразования значительно возрастает. Если

для 56-битового ключа нужно выполнить максимально 256 попыток лобовой атаки и при этом требуется хранить в памяти компьютера 256 64-титовых блоков или 1017 байт, то при двойном DES число попыток взлома возрастает до 256 , а объем требуемой памяти – до 2115 или примерно до 5\*1033 байт.

Следующий вариант объединения предложен У. Тачменом. Он преду-сматривает выполнение формально трех DES (*тройной* *DES* или 3DES). Один из вариантов такого подхода предполагает следующие операции: от-правитель сначала шифрует сообщение первым ключом, зачем расшифровы-вает результат вторым ключом и, наконец, опять шифрует первым ключом:

* *= EK1(DK2(EK1(M))),*

получатель соответственно расшифровывает сообщение первым клю-чом, зашифровывает вторым, расшифровывает первым:

* 1. *= DK1(EK2(DK1(C))).*
* литературе такой режим часто называют *зашифрование-*

*расшифрование-зашифрование* (*Encrypt-Decrypt-Encrypt*, EDE).

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

* *= EK3(DK2(EK1(M))), M = DK1(EK2(DK3(C))).*

Существуют и иные схемы многократного шифрования.

1. заимствован из книги: Б. Шнайер. Прикладная криптография.- М.:Триумф.-2003.-816 с

57

***5.3 Другие блочные алгоритмы***

Кратко охарактеризует некоторые из известных блочных алгоритмов симметричной криптографии.

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

**Алгоритм IDEA** (International Data Encryption Algorythm **–** междуна-родный алгоритм шифрования данных). Работает с 64-битовыми блоками данных и 128-битовым ключами. Является подстановочно-перестановочным алгоритмом. Блок данных делится на четыре 16-битовых блока, являющиеся входами первого раунда. Весь алгоритм состоит из 8 раундов. В каждом ра-унде 4 блока подвергаются операциям XOR, сложениям и умножениям друг с другом и шестью 16-битовыми подключами. Между раундами второй и тре-тий блоки данных меняются местами. В конечной операции четыре 16-битовых блока данных объединяются с 16-битовыми подключами.

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

**Алгоритм ГОСТ 28147-89**.Разработан в Советском Союзе.Работает

* 64-битовыми блоками данных и 256-битовыми 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.
  + *i*-oм раунде выполняются следующие операции:

|  |  |
| --- | --- |
| *Li=Ri-1,* |  |
| *Ri=Li-1 +f(Ri-1, Ki+1)*. | (5.2) |

Сравнив (5.1) и (5.2), вы не найдете отличий. Алгоритм предусматри-вает, как в DES, использование 8 S-блоков. 256-битовый ключ разбивается на восемь 32-битовых подключей, каждый из которых используется в одном ра-унде. Расшифрование выполняется в обратной последовательности. S-блоки, как и в алгоритме Lucifer, имеют по 4 входа и 4 выхода.

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

58

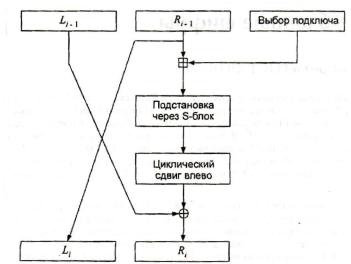


Рисунок 5.3. Один раунд алгоритма ГОСТ 28147-8

**Алгоритм Blowfish.** Автором является Б.Шнайер.Алгоритм не запа-тентован и его программная реализация общедоступна (приведена в прило-жении 2). Алгоритм создан для линий связи, где не предусмотрена частая смена ключа. Работает с 64-битовыми блоками данных. Предусмотрена воз-можность расширения ключа до 448 бит. Алгоритм состоит обычно из 16 раундов (может быть 12 или 20), в каждом из которых выполняются опера-ции подстановки и перестановки, зависящие от ключа. Используются только сложения и XOR над 32-битовыми словами. Каждый из 4 S-блоков хранит по

1. чисел.

Формально данный алгоритм, как и выше рассмотренные, представ-ляет собой *сеть Файстела*. 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 |

59

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

*Вопросы для контроля и самоконтроля*

1. Охарактеризовать общую схему алгоритма DES.
2. Охарактеризовать последовательность операций одного раунда алгоритма DES.
3. В чем сущность и польза лавинного эффекта в криптосистемах?
4. Описать формально сеть Файстела.
5. Пояснить сущность дифференциального и линейного методов криптоанализа.

***5.4 Асимметричная криптография***

***5.4.1 Основы асимметричной криптографии***

Две проблемы, связанные с практическим использованием симмет-ричных криптосистем (хранение и обмен ключевой 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 Хеллманов (Martin Hellman), и, независимо, Ральфом Мерклом (Ralph Merkle). В основу положена идея использовать ключи па-рами: один – для зашифрования (*открытый* или *публичный ключ*), другой – для расшифрования (*тайный ключ*). Таким образом, в отличие от симмет-ричных систем, ключи *К1* и *К2* (см. рис.3.1) отличаются. Несекретный ключ может передаваться по открытому каналу. Его знание не дает злоумышлен-нику возможности получить доступ к информации, содержащейся в сооб-щении.

У. Диффи и М. Хеллман в 1976 г. концепцию трансформировали в ал-горитм (в литературе соответствующий алгоритм называется *алгоритмом* *Диффи-Хеллмана*).Начиная с этого события,было создано множество алго-ритмов, основанных на концепции открытых ключей. Как правило, эти алго-ритмы основываются на решении трудных задач, которые мы рассматривали выше: вычисление дискретного логарифма и разложение чисел на множите-ли. 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*А*

* *В* могут воспользоваться этим алгоритмом для обмена ключевой информа-цией по открытым каналам. Предварительно стороны выбирают большие простые числа *n* и *g.*

60

Протокол обмена:

1. *А* выбирает случайное большое число *х*,вычисляет

*X = gx mod n*

* результат вычисления отсылает *В.*
  1. *В* выбирает случайное большое число *у*,вычисляет

*Y = gу mod n*

и результат вычисления отсылает *А.*

3. *А* вычисляет *k1* *= Yx* *mod n.*

4. *В* вычисляет *k2* *=* *Ху* *mod n.*

Таким образом, *k1* *= k2* *= gху* *mod 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*С*может перехватить открытое значе-ние, посылаемое от *А* к *В*, и послать вместо него свое открытое значение. За-тем он может перехватить открытое значение, посылаемое от *В* к *А*, и также передать вместо него свое открытое значение. Тем самым *С* получит общие секретные ключи с *А* и *В* 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 назна-чения.

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 зна-чению? Более формально задача формулируется так: дан набор значений *М1* *М*2,…, *Мn* и суммарное значение *S*.требуется вычислить значения *bn*„такиечто:

*S = b1 М*1 *+ b2 М2+... + bn Мп.*

Здесь *bi* может быть либо нулем, либо единицей. Значение *bi* = 1 озна-чает, что предмет *Мi* кладут в рюкзак, а *bi* = 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, исполь-зовав 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.

* 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 от-крытого текста соответствуют значениям *bi*, а шифртекст является получен-

61

ным суммарным весом. Пример текста, зашифрованного с помощью задачи укладки ранца, показан в табл. 5.9.

Таблица 5.9. Шифрование на основе укладки ранца

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Открытый текст | 1 1 1 0 0 1 | 0 1 0 1 1 0 |  | 0 0 0 0 0 0 | 0 1 1 0 0 0 |
| Ранец |  |  | 1 5 6 11 14 20 | |  |
| Вычисление шифра | 1+5+6+20= | 5+11+14= |  | 0 | 5+6= |
| Шифртекст | 32 |  | 30 | 0 | 11 |

Существуют две различные задачи укладки ранца: одна из них реша-ется легко и характеризуется линейным ростом трудоемкости, а другая - как принято считать, нет. Легкий для укладки ранец можно превратить в труд-ный. 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 применить легко разрешимый для укладки ранец, который предоставляет простой способ рас-шифрования сообщения. Тому, кто не знает закрытый ключ, придется попы-таться решить трудную задачу укладки ранца.

***5.4.2 Алгоритм RSA***

Вскоре после появления алгоритма укладки ранца Меркла был создан первый полноценный алгоритм с открытым ключом, который можно исполь-зовать 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 противо-стоит многочисленным попыткам криптоаналитического вскрытия. Безопас-ность алгоритма основана на трудоемкости разложения на множители (фак-торизации) больших чисел. Открытый и закрытый ключи являются функ-циями двух больших простых чисел, разрядностью 100 - 200 десятичных цифр или даже больше. Предполагается, что восстановление открытого тек-ста 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\* q*.Это первое число,входящее в ключ.
2. Вычисляется функция Эйлера (см. (4.3)):

(*n*) = (*p* – 1)(*q* – 1)*.*

62

1. Случайным образом выбирается простое число *e* – вторая часть ключа, которое удовлетворяет условиям: *e* (*n*); е и (*n*)) должны быть взаимно простыми числами.
2. Вычисляется число *d* *–* третья часть ключа, которое является об-рат- ным числу *e* (см. (4.4) и (4.5)), т. е.

*e \* d* 1 (mod(*n*)).

Пара чисел (*e*, *n*) делается открытым ключом и помещается в об-щедоступный справочник (база данных), а о числах *p*, *q* можно забыть (но это также является тайной информацией), Пара (*d*, *n*) – секретный ключ (по-нятно, 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, отме-тим 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.

Использование ключа.

*Зашифрование.* Если шифруется сообщение *М*,состоящее из *r* бло-

ков: *m1, m2* *, …,* *mi* *,…,* *mr,* то шифртекст *С* будет состоять из такого же числа (*r*) блоков, представляемых числами:

|  |  |
| --- | --- |
| *ci = (mi)e mod n.* | (5.3) |
| *Расшифрование*.Для расшифрования каждого зашифрованного блока | |
| производится вычисление вида: |  |
| *mi = (ci)d mod n.* | (5.4) |

Рассмотрим примеры.

*Пример 5-1*.Генерация ключа.Принимаем *p* = 11, *q* = 5.

Вычисляем *n* = 11 5 = 55.

Определяем функцию Эйлера:  (55) = (11–1) (5–1) = 40.

Выбираем ключ зашифровывания *e* = 7, который удовлетворяет усло-виям 7 40; НОД (7, 40) = 1.

Определяем *d* *–* ключ расшифровывания – из уравнения

7*d* 1 (mod 40)*.*

Рассмотрим способ нахождения *d*.

Для решения уравнения 7 *d* 1 (mod 40) используем алгоритм Евк-

лида:

40 = 7 5 + 5;

7 = 5 1 + 2;

5 = 2 2 + 1;

2 = 1 2 + 0.

Обратная подстановка дает:

63

1 = 5 – 2 2 = 5 – (7 – 5 1)2 = 5 3 + 7(–2) =

* (40 – 7 5)3 + 7(–2) = 40 3 + 7(-17). Поскольку -17 23 (mod 40), то *d* = 23.

Зашифровываем сообщение *М*=15 (сообщение состоит из одного бло-

ка), используя (5.3):

*C* 157(mod 55)5.

Для расшифрования *С* воспользуемся (5.4):

*M* 523(mod 55) =15.

*Пример 5-2*.Пусть *p*=47и *q*=71**,**тогда *n =* 3337.

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

*d= 79-1 mod 3220 = 1019.*

Пусть сообщение имеет следующий вид: *М* = 688232687 и длина бло-

ка равна 3:

*m1*=688, *m2*=232, *m3*=687.

Зашифрование**:** *с1*= 68879 (mod 3337)=1570 и т.д.

Расшифрование: *m1*=15701019 (mod 3337)=688 и т.д.

*Пример 5-3*.Необходимо зашифровать/расшифровать сообщение М=«ВСА» при *p*=11 и *q*=3.

Вычисляем элементы ключа: *n =*33;  (*n*)= 10\*2 = 20; *е* = 7 (взаимно простое с 20); и *d=*3.

Сообщение представляем в числовой форме: *m1*=2, *m2*=3, *m3*=1 (номе-ра соответствующих букв в алфавите).

Зашифрование: *с1*= 27 (mod 33)= 29; *с2*= 37 (mod 33)= 9; *с3*= 17(mod 33) = 1.

Поскольку вычисления ведутся по модулю 33, целесообразно предпо-ложить, 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 - со-ответственно «1» - «6». В соответствии с этим шифртекст сообщения можно представить в виде чисел (*С*=290901) либо в виде символов условного алфа-вита: С= «3IА».

Расшифрование: *m1*= 293 (mod 33) = 2 (соответствует букве «В»); *m2*=93(mod 33)=3 («С»); *\m3*=13(mod 33)=1 («А»).

**Криптостойкость алгоритма**.Зависит от трудоемкости решенияпроблемы разложения на множители больших чисел. *―*Лобовой метод*‖* вскрытия системы RSA заключается в нахождении числа *d*, обратного е по модулю (*n*). Это легко сделать, если известны числа *p* и *q*. Отметим, что математически не доказано, что для восстановления сообщения по шиф-ртексту и по значению открытого ключа нужно разложить *n* на множители. Тем не менее, именно этот подход является наиболее очевидным. Известно16, что факторизованы числа, длиной более 512 бит. Значит, нужно выбирать *n*

1. смотрите, например, [www.rsa.com](http://www.rsa.com/) (или воспользуйтесь запросом по адресу challenge-info@rsa.com)

64

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 вероят-ность взлома шифра значительно возрастает.

Другое направление взлома шифра. Криптоаналитик располагает шифртекстом *С=* *Ее(М)***.** Цель – восстановить *М* (расшифровать *С*). Для этого он шифрует известным открытым ключом произвольное сообщение *М’*. По-лучает *С’*. Если *С=С’***,** то получено *М* (*M=M’*), в ином случае шифруется дру-гое сообщение *М’’* и т.д.

* 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 при-мерно в 100 раз.

***5.4.3 Алгоритм Эль-Гамаля***

Данный алгоритм является альтернативой алгоритму RSA и, при рав-ном значении ключа, обеспечивает ту же криптостойкость. Стойкость алго-ритма Эль-Гамаля основана на трудности вычисления дискретных логариф-мов.

Предварительный этап – *генерация ключа*, состоящего из четырех чи-

сел.

1. Выбирается простое число *р* и два случайных числа, меньших, чем *р*:числа *х* и *g*.
2. Далее вычисляется
   1. *= gx mod p*.

Открытый ключ: *y, g* и *р*; тайный ключ: *х*. Использование ключа.

* отличие от алгоритма RSA, рассматриваемый алгоритм предусмат-ривает использование дополнительного параметра, который обозначим *k*. Ис-пользование этого параметра снижает вероятность взлома шифра. Это число является секретным и должно быть взаимно простым с *р-1*.

*Зашифрование.* Если шифруется сообщение *М*,состоящее из *r* бло-

ков: *m1, m2* *, …,* *mi* *,…,* *mr,* то шифртекст *С* будет состоять из такого же числа (*r*) блоков, представляемых *парой чисел*:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *ai = gk mod p*, | | | | | (5.5) |  |
| *b* | *i* | *= (* | *y* | *k m )mod p.* | (5.6) |  |
|  |  | *i* |  |  |

*Расшифрование*.Для расшифрования производится вычисление вида:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *m = (b* | | *)/(* | *ai*) | *х mod р.* | (5.7) |  |
| *i* | *i* |  |  |  |  |
|  |  |  |  |  | 65 |  |

Так как *(ai*)*х* в силу (5.5) можно заменить на *gkх* *mod p* и далее c учетом того, что *y = gx* *mod p,* можем получить подтверждение справедливости (5.7):

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *(b* | *)/(* | *ai*) | *х≡ (* | *k m )/ (* | | *х (mod p) ≡(* | | *kx m )/* | *kx (mod p)= m mod p.* | |  |
| *i* |  | *y* | *i* | *ai*) | | *g* | *i* | *g* | *i* |  |
| Справедливо также | | | | | |  |  |  |  |  |  |
|  |  |  |  |  | *mi* **= (***bi* \*(*ai*)*p-1-x*) *mod* p. | | | | | (5.8) |  |

Рассмотрим пример.

*Пример 5-4*.Необходимо выполнить прямую и обратную криптогра-фическую процедуру, если *М* = «Абрамов».

Выбираем *р*= 23, *g=*5 *и х=*3. Вычисляем *y = gx* *mod p=* 53 *mod 23*= 10. Предполагаем, что исходный текст строится на основе алфавита,

мощностью *р*. В этом случае представление сообщения *М* в виде цифровой последовательности будет выглядеть так: 01 02 18 01 14 16 03, т.е *m1* *=*1*, mi*

*=*2*, m3 =*18и т.д.

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

*ai = gk mod p* **=**57mod 23 = 17.

Используем (5.6) для вычисления второй части шифртекста: для первой буквы (А): *b1* *= (yk1* *\*М1) mod p* **=** (107\*1) mod 23 = 14, для второй буквы (б): *b2* *= (yk2* *\*М2) mod p* **=** (107\*2) mod 23 = 05, для третьей буквы (р): *b3* *= (yk3* *\*М3) mod p* **=** (107\*18) mod 23 = 22

и т.д.

Шифртекст (пары *ai* и *bi*): *С*= 1714 1705 1722…. .

Как видим, длина зашифрованного сообщения действительно удвои-

лась.

*Расшифрование*.

Используем (5.7): *m1* *= (b1* *((a1)x)-1) mod p* **=** (14 (( 17)3)-1) mod 23 == (14 \* (4913) -1) mod 23 = ((14 mod 23 )\* ((4913) -1 mod 23 )) mod 23 = = (14\*5) mod

23 = 70 mod 23 = 1**,** т.е *m1* **=** «А».

При этом мы приняли во внимание: (4913) -1) mod 23 =5, т.к. 4913 \* 5 mod 23 = 1.

Или используем (5.8):

*m1* **=** *(b1 (a1)p-1-x) mod p* =(14\* (17)23-1-3) mod 23= (14\* (17)19) mod 23 == (14\* 239072435685151324847153) mod 23 =1.

Аналогично получаем остальные значения зашифрованного сообще-

ния.

Если абоненты *А* и *В* по согласованию выбрали одинаковые значения *р*

* *g*,то для обмена зашифрованными сообщениями они могут воспользоватьсяидеей, реализованной в вышерассмотренном алгоритме Диффи-Хеллмана (в литературе иногда указывается, что алгоритм Эль-Гамаля подпадает под дей-

66

ствие патента, полученного У. Диффи и М. Хеллманом). Сторона *А* генериру-ет секретный ключ *хА* *< p* и вычисляет открытый ключ:

|  |  |
| --- | --- |
| *yА = g хА mod p*, | (5.9) |

сторона *В* выбирает число *хВ* *< p* и с его помощью зашифровывает пере- да-ваемое сообщение *М* следующим образом:

|  |  |
| --- | --- |
| *yВ = g хВ mod p*, | (5.10) |
| *С* = *М* ((*yА*) *хВ mod p*)). | (5.11) |

* последнем случае суммирование ведется по модулю два, а сумми-руемые числа представляются в двоичном виде.

Предварительно стороны обмениваются открытыми ключами *yА* и *yВ.* Понятно, что закрытые ключи (*хА* и *хВ*) каждая сторона хранит в тайне.

Сторона *А*, получив сообщение в виде *yВ* и *C*, восстанавливает его:

|  |  |
| --- | --- |
| *М* = ((*yВ) хА mod p*) *С*. | (5.12) |

*Пример 5-5*.Пусть *p* = 11; *q* = 3; *хА* = 7; *хВ* = 4; *М* = 6.

Открытый ключ, посылаемый стороне *В*, равен (см. (5.9)):

*yА* 37(mod 11)2187(mod 11)9.

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

*yВ* 34(mod 11)81 (mod 11)4;

для вычисления *С* предварительно определим второе слагаемое в (5.11):

(*yА*) *хВ* *mod p*) = 94 (mod 11) = 5 (в двоичном виде – 101).

И, наконец, *C* = 110 101 = 011, где двоичным представлением *М* бу-дет последовательность 101.

Сторона *А*, получив зашифрованное сообщение в виде *yВ* и *C*, расшифро-

вывает его, используя (5.12): 47 (mod 11)и далее десятичное число 6 – М = 6).

*Вопросы для контроля и самоконтроля*

1. На чем основана идея обмена ключевой информацией Диффи и

Хеллмана?

1. Сформулировать задачу укладки ранца и показать примеры ее

решения.

1. Зашифровать и расшифровать сообщения (свои фамилию, имя, отчество) с использованием алгоритма Эль -Гамаля для ключевых значений, представленных следующей таблицей.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | *p* | *g* | *хА* | *хВ* | № | *p* | *g* | *хА* | *хВ* |  |
| варианта | варианта |  |
|  |  |  |  |  |  |  |  |  |
| 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 |  |

67

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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***

* 1985 году Н. Коблиц (N. Koblitz) и (независимо) В. Миллер (V. Miller) предложили использовать эллиптические кривые для создания крип-тосистем с открытым ключом.

Их безопасность, как правило, основана на трудности решения задачи дискретного логарифмирования в группе точек эллиптической кривой над *конечным полем*.Этим и обусловлена их высокая криптостойкость по срав-нению с другими алгоритмами.

*Эллиптические кривые – математический объект, который может быть определен над любым полем*.В криптографии обычно используютсяконечные поля. Для точек на эллиптической кривой вводится операция сло-жения, которая играет ту же роль, что и операция умножения в криптосисте-мах RSA и Эль-Гамаля. Еще одним преимуществом криптосистем на эллип-тических кривых является высокая скорость обработки информации. Напри-мер, уровень стойкости, который достигается, скажем, в RSA при использо-вании 1024-битовых ключей, в системах на эллиптических кривых реализует-ся при том же параметре длиной 160 бит.

Криптосистемы на эллиптических кривых, как, впрочем, и другие криптосистемы с открытым ключом, нецелесообразно применять для шиф-рования больших объемов данных. Но зато их можно эффективно использо-вать для систем цифровой подписи и ключевого обмена. С 1998 года ис-пользование 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 – ГОСТ Р34.10–2001 был принят в России.

***5.4.4.1 Представление и описание эллиптической кривой***

Для описания эллиптической кривой используется *алгебраическая*

*геометрия****17***.

1. в некотором приближении алгебраическая геометрия занимается алгебраическими множествами и алгеб-раическими отображениями

68

*Эллиптическая кривая* (ЕС–EllipticСurve)–это не эллипс.Она на-зывается так называются потому, что описывается кубическим уравнени-ем, подобным тем, которые используются для вычисления кривой эллип-са.

* общем случае кубические уравнения для эллиптических кривых имеют вид

|  |  |
| --- | --- |
| *у2 + аху + by = х3 + сх2 + dx + е*, | (5.13) |

где *а*, *b*, с, *d*, и *е* являются действительными числами, удовлетворяющими некоторым простым условиям. Такие уравнения называются еще *уравнения-ми третьего порядка*,поскольку в них наивысший показатель степени равентрем.

* другой стороны, эллиптическая кривая – это набор точек (*x, y*), удовлетворяющих уравнению формы (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 состоитиз конечного множества *F* и двух бинарных операций (сложения и умноже-ния).

*Порядком поля* называется количество элементов в поле(во множест-

ве *F*).

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

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

|  |  |
| --- | --- |
| *у2 = х3 + aх + b (mod p),* | (5.14) |

при этом константы должны удовлетворять условию:

4*a3+*27*b2* *≠ 0 (mod p).*

*Пример 5-6.* Рассмотрим эллиптическую кривую,описываемую урав-нением *у2* *=* *х3* *+* *3х* *+ 2* над полем GF(*5*). На этой кривой, в частности, лежит точка (*х* =1, *у* =1), так как

12 = 13 + 3\*1 + 2 *(mod 5).*

Поля характеристики 2, GF(*2m*), называют также *бинарными конечны-ми полями*.У эллиптических кривых над полемGF(*2m*)есть одно важное пре-имущество, элементы поля GF(*2m*) могут быть легко представлены в виде n-битовых кодовых слов, это позволяет увеличить скорость аппаратной реали-зации эллиптических алгоритмов.

69

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

|  |  |  |
| --- | --- | --- |
| • | суперсингулярная кривая: |  |
|  | *у2 + аy = х3 +bx + c,* | (5.15) |
| • | несуперсингулярная кривая: |  |
|  | *у2 + аyx = х3 +bx2 + с.* | (5.16) |

Определение эллиптической кривой включает также некоторый эле-мент, называемый *несобственным элементом* (а также *бесконечным эле-ментом*,или *нулевым элементом*).

**Таким образом, помимо точек, принадлежащих кривой (коорди-наты 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 *Е23(1, 1)* будет означать, что мы ис-пользуем уравнение кривой вида

*у2 = х3 + х + 1 (mod 23).*

**Эллиптическая кривая или множество *Ep*(*a*, *b*) состоит изо всех точек (*х,* *у*), *х* 0, *у* *<* *р*, удовлетворяющих уравнению (5.14), и точки в бесконечности *О****.*

Количество точек в *Ep*(*a*, *b*) будем обозначать *#Ep*(*a*, *b*) = *N.*

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

Эта величина имеет большое значение для криптографических при-ложений эллиптических кривых. Чтобы использовать эллиптическую кри-вую в криптографии необходимо знать ее порядок. Это обуславливается тем, что от порядка кривой зависит криптостойкость системы на основе со-ответствующей кривой (уравнения кривой).

***Арифметические операции в эллиптической криптографии* произ-водятся над точками кривой.**

Основной операцией является сложение. *Сложение двух точек* легко представить графически (см. рис. 5.4). Вид уравнения указан на рисунке.

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 относительногоризонтальной оси.

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

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

70

* пусть *P* и *Q* – две различные точки эллиптической кривой (рис.5.4), и *Р* не равно *Q*. Проведем через *P* и *Q* прямую. Она пересечет эллиптическую кривую еще только в одной точке, называемой –*R*. Точка –*R* отображается относительно оси *Х* в точку *R,* равную сумме точек *P* и *Q* ;

**Закон сложения точек эллиптической кривой**: ***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* = (*х, –у*), то *R* + (*х,* *у*) = *О*. Точка (*х,* *у*) является отрицательным зна-чением точки *R* и обозначается –*R*;

таким образом, **по определению** ***R + (–R) =*** ***О***;

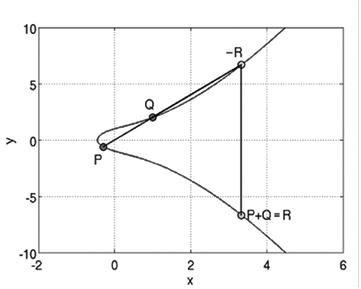


Рисунок 5.4. Пояснение к операции сложения двух точек

эллиптической кривой *у* *=* *х3* *+ 2\*х* *+1*

* + если *O* – нулевой элемент, то справедливо равенство *O* = –*O*, а

**для любой точки *P* эллиптической кривой имеем *P + O = P***;

* + чтобы сложить точку *Р* с ней самой, нужно провести касательную к кри-вой в точке *Р*.

**Закон удвоения точки *Р*: *P + P = 2\*P***;

* **умножение точки *Р* на целое положительное число *k*** определяетсякак сумма *k* точек *Р:* ***k\*P = P + P + P +*** ***…*** ***+ P***;

скалярное умножение осуществляется посредством нескольких ком-бинаций сложения и удвоения точек эллиптической кривой. Например, точка 25\**P* может быть представлена, как

25\**P* = 2\*(2\*(2\*(2\**P*)) + 2\*(2\*(2\**P*))) + *P*.

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

71

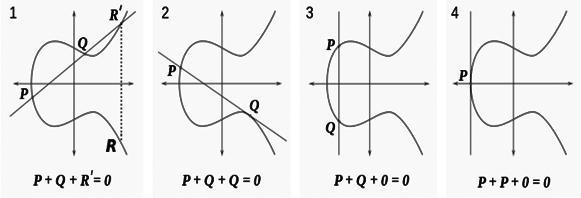


Рисунок 5.5. Примеры некоторых арифметических операций над точ-ками эллиптической кривой

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, проведѐнной через *P* и *Q (*пер-вый пример слева); если прямая, которая используется для выполнения опе-рации сложения точек, является касательной к кривой в точке, то такая точка считается дважды (второй слева пример) ; если же прямая параллельна оси ординат, третьей точкой будет точка в бесконечности (третья и четвертая ил-люстрации).

**Идея, надежность и криптостойкость эллиптической криптогра-фии напрямую связана с операцией умножения точки на целое число**:

**задача вычисления дискретного логарифма на эллиптической кривой, заключающаяся в отыскании целого числа *х* по известным точкам *P* и *Q***

* ***х\*P*,является трудноразрешимой.**

Если *Р* *=* *(х1,* *у1)* и *Q* = (*х2,* *у2*), то *Р* *+ Q =* *(х3,* *у3)* определяется в соот-ветствии с правилам:

|  |  |
| --- | --- |
| *x3 2 – х1 – х2 (mod p);* |  |
| *у3 (х1 – х3) – у1 (mod p),* | (5.17) |

где ** *у2–у1)/(* *х2–* *х1)*, если *Р* *≠* *Q* и ** *(х1)2+а)/2* *у1,* если *Р* *= Q.*

Из этого следует, что число **– угловой коэффициент секущей, проведенной через точки *Р* *=* *(х1,* *у1)* и *Q* = (*х2,* *у2*). При *Р* = *Q* секущая пре-вращается в касательную, чем и объясняется наличие двух формул для вы-числения **

*Пример 5-7*.В случае *Е*23(1, 1)–соответствует вышеприведенномууравнению *у2* *=* *х3* *+* *х* *+ 1 (mod 23)* – для *Р* = (13, 7) имеем –*Р* = (13, –7). Но

–7 *mod* 23 16, таким образом, –*Р* = (13, 16), т.е. *х= 13,* *у*=16.

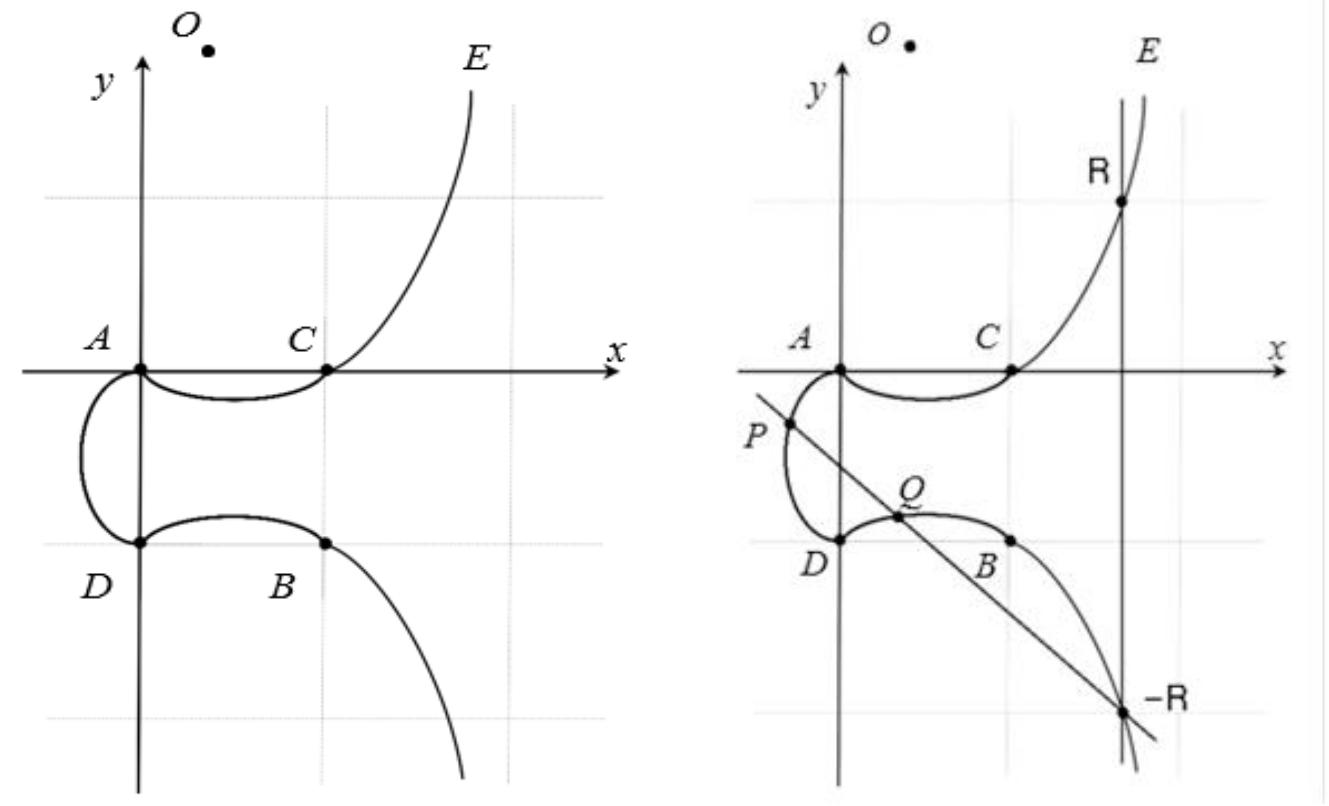
Здесь *следует вспомнить,* *что значение некоторого целого отрица-тельного числа (–k) по модулю (р) вычисляется следующим образом*:

72

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | (–*k*) *mod* *р* *=* – *(k mod* *р) + p.* | (5.18) |
|  | Возьмем более общий пример. | |  |
|  | *Пример 5-8*.Рассмотрим эллиптическую кривую | | *Е* (рис.5.6),соответ- |
|  |  |  |  |
| ствующую уравнению | | |  |
|  |  | *y*2+ *y* = *x*3– *x*2. | (5.19) |

На этой кривой лежат только четыре точки, координаты которых явля-ются целыми числами. Это точки *A*(0, 0), *B*(1, –1), *C*(1, 0), *D*(0, –1). На плос-кости существует бесконечно удаленная точка *O* *E*, в которой сходятся все вертикальные прямые.

Выполним операцию сложения точек *Р,* *Q* *E* (рис. 5.7). Для этого проведем прямую линию через точки *Р* и *Q*, найдем третью точку, *-R* пе-ресечения этой прямой с кривой *Е*. Далее проведем через точку *-R* верти-кальную прямую до пересечения с кривой *Е* в точке *R*, которая будет иско-мой суммой: *Р* *+ Q = R*.



|  |  |
| --- | --- |
| Рисунок 5.6. Группа из пяти точек | Рисунок 5.7. Графическое пред- |
| эллиптической кривой *Е*; | ставление сложения точек: |
| *О* –бесконечно удаленная точка | *Р + Q = R* |

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

*A + A = B; A + B = C; A + С = D; A + D = O*,

или

73

*2A = B; 3A = C; 4A = D; 5A = O; 6A = A.*

Еще раз отметим, что важнейшим параметром эллиптической кривой, определяющим возможность ее использования в криптографии, является *ко-личество точек* этой кривой: *#Ep*(*a*, *b*) = *N*.

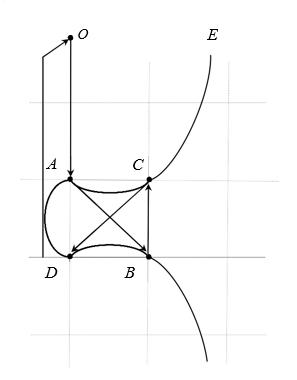


Рисунок 5.8. *Абелева группа* ({*A, B, C, D, O*}) на кривой *E*

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

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

1. *a + b = b + a,* ∀ *a, b ∈ A* (коммутативность);
2. *(a + b ) + c = a + (b + c ),* ∀ *a, b ∈ A* (ассоциативность);
3. во множестве *A* существует такой элемент *O* (нуль), что *a + O = a*,∀ *a ∈ A*;
4. для любого элемента *a* *∈* *A* существует такой элемент *(–a)* *∈* *A* (противопо-ложный элемент), что *a* *+ (−a) =* *O*.

Нетрудно понять, что, если вместо *a* и *b* подставить точки кривой *Е* (все точки которой вместе с точкой *О* и образуют множество *А*), то мы полу-чим подтверждение корректности всем рассмотренным нами примерам.

*Пример 5-9*.Рассмотрим кривую *E*7(2, 6): *y*2*x*3+ 2*x* + 6 (*mod* 7),т.е.коэффициентами уравнения являются: *а*=2 и *b*=6. Проверим выполнение ус-ловия несингулярности кривой (см. (5.14)):

74

(4 \* 23 + 27 \* 62) mod 7 3 (mod 7) 0.

Итак, данная кривая несингулярна. Найдем какую-нибудь (случайную) точку на *E*7(2, 6). Пусть *х* = 5, тогда

*y*253+ 2 \* 5 + 6 (*mod* 7)(125 + 10 + 6) *mod* 71 *mod* 7

* *у* 1 (*mod* 7)или(с учетом(5.18)): *у* –16 (*mod* 7),поскольку(-1)2= 1.

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

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

** *(х1*)*2+а)/*2\* *у1* (*mod* 7)= (3\*52+ 2)/ 2\*1(*mod* 7)0 (*mod* 7)

и далее:

*x*3= 0–2 \* 5 (mod 7)4 (*mod* 7);

у3 = 0 (5 – 4) – 1 (mod 7) 6 (*mod* 7).

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

*y*2*x*3+ 2*x* + 6 (*mod* 7).

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



              (mod 7) (*mod* 7) 2 (*mod* 7); *x*3= 22–5–4 (mod 7)2 (*mod* 7);

у3 = 2 (5 – 2) – 1 (mod 7) 5 (*mod* 7).

Итак, найдены четыре точки. Еще раз подчеркнем: для криптографического использования кривой важно знать, сколько всего точек в множестве *E*7(2, 6).

*Пример 5-10*.Пусть *р* = 23.Рассмотрим эллиптическую кривую Е:

*y*2= *x*3+ *x* + 1 (*mod* 23),

т.е. Е23(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 *Р* = (3, 10) и *Q* = (9, 7). Найдем *Р* + *Q* и 2\*Р*.*

Пусть *Р* *+ Q* = (*х3,* *у3*), тогда при

(*mod* 23) (*mod* 23)

имеем:

*x3* = 121–3–9 (*mod* 23) = 109 (*mod* 23)17 (*mod* 23);

*у3* = 11(3 + 6)–10 (*mod* 23) = 89 (*mod* 23)20 (*mod* 23).

75

Таким образом, *Р* *+* *Q* = (17, 20).

Найдем теперь точку 2\**Р* = *Р* *+* *Р* = (*х3,* *у3*) – с формальной точки зре-ния это также будет третья точка. Для этого случая

\*(*mod* 23) (*mod* 23)

и с учетом (5.18) для вычисления координаты *у*3:

*x*3= 36–6 = 307 (*mod* 23);

*у*3= 6\*(3–7)–10 =–3412 (*mod* 23).

Таким образом, 2\**Р* = (7, 12).

При последовательном выполнении сложения ***k\*P*** ***=***

* ***P + P + P + … + P*** на каждом шаге будет получаться точка,которая такжедолжна принадлежать *Ep*(*a*, *b*). В силу того, что эллиптическая группа содер-жит конечное множество точек, наступит такой момент, что для некоторых результатов вычислений будет выполняться равенство ***q\*Р*** ***=*** ***О*** (см. пример 5.8, где *5\*A = O,* т.е. здесь *q*=5).

**Определение 5.3. Наименьшее значение числа *q*, для которого вы-полняется равенство *q\*Р* *=* *О*, называется *порядком точки Р***.

**Определение 5.4. Порядок группы точек эллиптической кривой равен порядку группы точек ЭК (включая точку *О*)**.

Для ЭК *Ер* *(а,b)* порядок *m* группы точек должен удовлетворять нера-

венству:

|  |  |
| --- | --- |
| *p + 1 – 2\*(p)½ ≤ m ≤ p + 1 + 2\*(p)½* . | (5.20) |

* таблице 5.10 приведены результаты последовательного сложения для некоторых точек кривой вида *y2* *= x3* *+1*.

Таблица 5.10. Результаты выполнения операции *Р*+ …+*Р* для кривой *Е5(0,1)*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **+** | **(0,1)** | | | **(0,4)** | | | **(2,2)** | | | **(2,3)** | | | **(4,0)** | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ***Р*** | (0,1) | | | (0,4) | | | (2,2) | | | (2,3) | | | (4,0) | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| ***2Р*** | (0,4) | | | (0,1) | | | (0,4) | | | (0.1) | | |  | *О* |  |
|  |  |  | |  |  |  |  |  |  |  |  |  |  | | |
| ***3Р*** |  | *О* |  |  | *О* |  | (4.0) | | | (4.0) | | | (4,0) | | |
|  |  | | |  | | |  |  |  |  |  |  |  |  | |
| ***4Р*** | (0,1) | | | (0,4) | | | (0.1) | | | (0,4) | | |  | *О* |  |
|  |  | | |  | | |  |  |  |  |  |  |  | | |
| ***5Р*** | (0,4) | | | (0,1) | | | (2.3) | | | (2.2) | | | (0.4) | | |
|  |  |  | |  |  | |  |  |  |  |  |  |  |  | |
| ***6Р*** |  | *О* |  |  | *О* |  |  | *О* |  |  | *О* |  |  | *О* |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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 *р*, простым числом. Если умножать эту точку на

76

числа меньшие, чем порядок точки, каждый раз будут получаться различные точки кривой.

**Определение 5.3. Генерирующая (базисная) точка эллиптической кривой – это такая точка *G* на эллиптической кривой, для которой ми-нимальное значение *n*, такое, что *n\*G = O*, является *очень большим про-стым числом*.**

Именно произведение точки *G* на большое случайное число исполь-зуются как открытый параметр, а число **–** тайный, например, при реализа-ции алгоритма Диффи-Хеллмана или ЭЦП (см. р. 12 настоящего пособия) на эллиптических кривых.

Выбор генерирующей точки обусловлен тем, чтобы ее порядок *q* был достаточно большим: обычно 2254 < *q* < 2256. В таблице 5.11приведены ре-зультаты выполнения операции *х\*G* для кривой *E23(10,10)* с генерирующей точкой *G=Р(5,1).*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Таблица 5.11. | | Результаты выполнения операции *х\*G* для | | | | | | | | | кривой | |
| *E23(10,10)* с генерирующей точкой *G=Р(5,1)* | | | | | |  |  |  |  |  |  |  |  |
| *х* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |  |  |  | 9 |
| *х\*G* | (5,1) | (8,21) | (11,5) | (10,11) | (12,8) | (7,20) | (15,19) | (9,1) | | | |  | (9,22) |
| *х* | 10 | 11 | 12 | 13 | 14 | 15 | 16 |  | 17 | |  |  | 18 |
| *х\*G* | (15,4) | (7,3) | (12,15) | (10,12) | (11,8) | (8,2) | (5,22) |  | *О* |  | |  | (5,1) |

Как видно, для выбранной генерирующей точки ее порядок равен 17, так как 17 является минимальным целым простым числом, для которого вы-полняется равенство *n\*Р* *=* *О*.

***5.4.4.2 Рекомендации по выбору параметров эллиптической кривой***

Рассмотрим основные рекомендации по выбору параметров эл-липтической кривой, предназначенной для решения криптографических за-дач, а именно: задач по выбору коэффициентов *a*, *b* и модуля *р*. Основным критерием выбора является вероятность осуществления атак на криптоси-стему на основе эллиптической кривой.

Рассматриваемая ниже стратегия считается наиболее надежной с

точки зрения обеспечения криптостойкости системы.

Рассмотрим по шагам процесс формирования кривой.

Шаг 1. Выбираем случайно простое число *р*. Его длина в

битах ( *t =* log *p* + 1; понятно, что *t* должно быть целочисленным с округ-лением в большую сторону) должна быть такой, чтобы сделать невозмож-ным применение общих методов нахождения логарифмов на кривой. Вели-чина *t* в настоящее время принимается не менее 256 бит.

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

*a*, *b* (mod *p*)0 и 4*a*3 + 27*b*2(mod *p*) 0 *.*

77

Следует обратить внимание на то, что при вычислении композиции точек па-раметр *b* нигде не фигурирует. Поэтому для повышения эффективности сче-та рекомендуют случайно выбирать только *b*, а *а* принимать равным не-большому целому числу (например, - 3).

Шаг 3. Определяем число точек *N* на кривой, равное *#Ep*(*a*, *b*). Важно, чтобы число *N* имело большой простой делитель, *q*, а лучше всего

– само было простым числом: *N = q.* Если поиск кривой с *N = q* занимает слишком много времени, то можно допустить *N = h q,* где *h* *–* небольшое число. Еще раз следует подчеркнуть, что стойкость криптосистемы на эл-липтической кривой определяется не модулем *р*, а числом элементов *q* в подмножестве точек кривой. Но если множитель *h* – небольшое число, то *q* является величиной того же порядка, что и *р*. Если *N* не соответствует предъявляемым требованиям, то необходимо вернуться к шагу 2.

Шаг 4. Проверяем, выполняются ли неравенства (*рk* – 1) *mod q* 0 для всех *k*, 0 < *k <* 32. Если нет, то возвращаемся к шагу 2. Эта проверка предотвращает возможность некоторые виды атак и исключает из рассмот-

рения так называемые суперсингулярные кривые и кривые с *N* = *р* – 1.

Шаг 5. Проверяем, выполняется ли неравенство *q* *р*. Если нет, то возвращаемся к шагу 2. Д ля кривых с *q =* *р*, которые называются аномаль-ными, существуют эффективные методы вычисления логарифмов, т.е. взлома шифра.

Шаг 6. Необходимая для криптографических приложений кривая уже получена. Имеем параметры *р*, *а*, *b*, количество точек *N* и размер подмноже-ства точек *q.* Чтобы получить случайную точку на кривой, берем случайное

число *х* < *р*, вычисляем *е* (*x*3 *+ ax + b*) mod *p* и пытаемся извлечь квад-ратный корень *y*= *e* mod *p*.Если корень существует,то получаем точку(*х*, *у*),в противном случае пробуем другое число х.

**Задача, которую решает криптоаналитик при использовании криптосистемы на базе эллиптических уравнений, относится к зада-чам дискретного логарифмирования на эллиптической кривой и формулируется следующим образом: даны точки *Р* и *Q* на эллиптиче-ской кривой порядка *N*, ( *N* – число точек на кривой). Необходимо найти единственное число *х*, такое, что *Р* *=* *х\*Q*. Величина *х* и являет-ся дискретным логарифмом от *Q* по основанию *Р****.*

***5.4.4.3 Система распределения криптографических ключей на основе эллиптической кривой***

* качестве примера использования эллиптических кривых для крип-тографических применений рассмотрим систему распределения (обмена) тайной ключевой информации. На основе кривых созданы и работают криптосистемы, использующие протоколы Эль-Гамаля, RSA и Шнорра18.

1. Например, схема Шнорра используется в стандарте цифровой подписи в РБ (см. гл. 10)

78

Наша система работает подобно вышеописанному алгоритму Диффи-Хеллмана (см. раздел 5.6.1), т.е. предназначена для передачи закрытой ин-формации по открытому каналу. Еще раз подчеркнем: стойкость рассмат-риваемого алгоритма определяется сложностью вычисления дискретного логарифма над эллиптической кривой *Eр(а,b)*. Действительно, вычисление точки *Q* как результата скалярного произведения целого положительного *х* на точку *G* является достаточно простой задачей. С другой стороны, вы-числение значения *х* на основании точек *Q* и *G* является вычислительно трудной задачей. Для больших значений *х* эта задача является практически неразрешимой. На этом факте и основана рассматриваемая криптографиче-ская система.

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

Сначала выбирается простое число *р* и параметры *a* и *b* для эллип-тической кривой. Это задает эллиптическую группу точек *Ер*(*a*, *b*)*.* Затем в *Ер*(*a*, *b*)выбирается генерирующая точка *G = (х, у*).При этом важно,чтобынаименьшее значение *q*, при котором выполняется условие *q\*G* = *О*, оказа-лось очень большим простым числом. Параметры *Ер*(*a*, *b*) и *G* криптосисте-мы являются параметрами, известными всем участникам. Обмен ключами между участники информационного процесса (*А* и *В*) можно провести по сле-дующему алгоритму.



1. Сторона *А* выбирает целое число *ka*, меньшее *q.* Это число будет тайным ключом стороны *А*. Затем *А* генерирует открытый ключ: *Ya* = *ka\*G* и

отсылает его стороне *В.*

Открытый ключ представляет собой некоторую точку из *Ер*(*a*, *b*)*.*

2 . Параллельно сторона *В* выбирает для себя тайный ключ *kb* и вычис-ляет открытый ключ: *Yb* = *kb\*G* и отсылает его стороне *А.*

3 . Сторона *А*, получив открытый ключ стороны *В*, генерирует секрет-ный ключ *Ка* = *ka* *\*Yb* = *К,* а сторона *В* соответственно генерирует секретный ключ *Кb* = *kb* *\* Ya*= *К.*

Не сложно установить, что обе стороны получили один и тот же ре-зультат, поскольку *ka* *\*Yb* = *ka* (*kb\* G*) = *kb* (*ka\* G*) = *kb\* Ya*.

*Пример 5-11*.Пусть *р* = 211; *G* = (2, 2); *Ер*(0,–4),что соответствует

кривой *у2* *=* *х3* *–* 4.

Стороны совместно выбирают *q=*241*.* Можно подсчитать, что 241\**G*

* *О.*

Тайным ключом стороны *А* является *ka* = 121, поэтому открытым ключом стороны *А* будет *Ya* = 121(2, 2) = (115, 48).

Тайным ключом стороны *В* будет значение *kb* = 203, поэтому откры-тым ключом стороны *В* будет *Yb* = 203(2, 2) = (130, 203).

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

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

79

*Вопросы для контроля и самоконтроля*

1. Записать общий вид уравнения эллиптической кривой.
2. Продемонстрировать графически порядок сложения двух точек

ЭК.

1. Что такое «точка *О*»?
2. Записать формально и представить графически «закон удвоения

точки» ЭК.

1. Чем определяется криптостойкость эллиптической криптогра-

фии?

1. Что такое «генерирующая точка» ЭК?
2. Сформулировать задачу дискретного логарифмирования над эл-липтическими кривыми.
3. Для ЭК *у2* *=* *х3* *+ aх* *+ b (mod 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 |

80

1. **Потоковые шифры**

***6.1. Общая характеристика потоковых шифров***

Как мы убедились, в блочном шифре из двух одинаковых блоков от-крытого текста получаются одинаковые блоки шифрованного текста, что, безусловно, является одним из недостатков алгоритмов. Избежать этого по-зволяют 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 *С* в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста, *М*.**

Потоковый шифр реализует другой подход к симметричному шифро-ванию, нежели блочные шифры.

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

Потоковые шифры преобразуют открытый текст в шифртекст *по од-*

*ному биту* (*символу*) *за операцию*. *Генератор потока ключей* выдает потокбитов: *k1, k2, …, ki*. Этот поток битов (иногда называемый *бегущим ключом*) и поток битов открытого текста, *m1*, *m2*, …, *mi,…*, *mz*, подвергаются операции сложения по модулю два (XOR), и в результате получается поток битов шифртекста:

|  |  |
| --- | --- |
| *ci* = *mi* *ki*. | (6.1) |
| При расшифровании, для восстановления битов открытого текста, | |
| операция XOR выполняется над битами шифртекста и тем же самым потоком | |
| ключей: |  |
| *mi* = *ci* *ki*. | (6.2) |

Как видим, справедливо: *ki* = *mi* *ci*.

Так как посимвольное шифрование при потоковом шифровании дан-ных не приводит к задержкам в криптосистеме, то важнейшее достоинство потоковых шифров – высокая скорость шифрования, соизмеримая со скоро-стью поступления входной информации.

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

Классический пример потокового шифра – ***шифр Вернама*** **или** ***одно-разовый блокнот***.Уже отмечалось,что если для гаммы последовательностьбитов выбирается случайно и длина гаммы равна, по крайней мере, длине сообщения, то взломать шифр невозможно. Но у данного режима шифрова-ния 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 по-

81

токовых шифров – реализовать концепцию одноразового блокнота, исполь-зуя секретный ключ меньшей длины, из которого для гаммы генерируется псевдослучайная числовая последовательность, похожая на случайную.

Безопасность системы полностью зависит от свойств генератора пото-ка ключей. Если этот генератор выдает бесконечную строку нулей, шиф-ртекст будет совпадать с открытым текстом и преобразование будет бес-смысленным. Если генератор потока ключей выдает повторяющийся, напри-мер, 16-битовый шаблон, криптостойкость системы будет пренебрежимо ма-ла. В случае бесконечного потока случайных битов криптостойкость потоко-вого шифра будет эквивалентна криптостойкости одноразового блокнота. Генератор потока ключей создает битовый поток, который похож на случай-ный, но в действительности обычно детерминирован и может быть с опреде-ленной вероятностью воспроизведен при расшифровании. Чем ближе выход генератора потока ключей к случайному, тем выше трудоемкость криптоана-литической 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, потоко-вые шифры больше подходят для аппаратной реализации.

Можно сказать, что зашифрование осуществляется *наложением гам-мы* (шифрование *гаммированием*).А сама гамма является ключом шифрова-ния. В практических поточных шифрах длина шифртекста много больше длины секретного ключа, а ключевая последовательность является псевдо-случайной

* имеет некоторый период. Очевидно, что если последовательность битов гаммы не имеет периода и выбирается случайно, то взломать шифр невоз-можно. Если бы генератор выдавал бесконечную последовательность битов, в которой каждый бит порождался независимо и с вероятностью 1/2 прини-мал значения 0 или 1 , то мы получили бы совершенно стойкий шифр. Но иметь ключ, равный по размеру шифруемым данным представляется про-блематичным. Поэтому потоковые шифры и вырабатывают выходную гамму на основе некоторого секретного ключа небольшого размера (например, 128 бит). С его помощью генерируется псевдослучайная гаммирующая последо-вательность (ПСГП). Она должна удовлетворять постулатам С. Голомба (все-го 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) долж-на 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 одина-ковое количество серий из "1" и "0".

Таким образом, ***основной задачей потоковых шифров является вы-работка некоторой последовательности (выходной гаммы) для шифро-***

82

***вания*,т.е.выходная гамма является ключевым потоком для сообщения**.

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

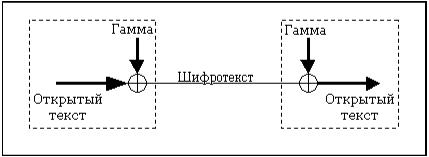


Рисунок 6.1. Схема потокового шифра

Допустим, что в режиме гаммирования для потоковых шифров при передаче по каналу связи произошло искажение одного символа шифртекста (или шифротекста; на эту аналогию мы обращали внимание выше). Очевид-но, что в этом случае все символы, принятые без искажения, будут расшиф-рованы правильно. Произойдѐт потеря лишь одного символа текста. А теперь представим, что один из символов шифртекста при передаче по каналу связи был потерян. Это приведѐт к неправильному расшифрованию всего текста, следующего за потерянным символом. Практически во всех каналах переда-чи данных для потоковых систем шифрования присутствуют помехи. Поэто-му для предотвращения потери информации решают проблему синхрониза-ции процедуры зашифрования и расшифрования текста. По способу решения этой проблемы шифрсистемы подразделяются на синхронные и системы с самосинхронизацией.

Таким образом, потоковые шифры классифицируются на:

*синхронные*;

*самосинхронизирующиеся (асинхронные)*.

Проанализируем кратко особенности каждой системы.

***6.2. Синхронные потоковые шифры***

**Определение 6.2***.* ***Синхронные потоковые шифры*** *(СПШ)*—шиф-ры, 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. По-теря символа шифртекста приведет к нарушению синхронизации между эти-ми двумя генераторами и невозможности расшифрования оставшейся части

83

сообщения. Очевидно, что в этой ситуации отправитель и получатель долж-ны повторно синхронизироваться для продолжения работы.

*Главное свойство СПШ – нераспространение ошибок*.Ошибки отсут-ствуют, пока работают синхронно шифровальное и дешифровальное устрой-ства отправителя и получателя информации. Перебои в работе системы назы-вают *рассинхронизацией*. Ее может вызвать несовпадение скоростей зашиф-рования/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 реинициализа-цией ключа обоими пользователями. Один из методов борьбы с рассинхро-низацией – разбить отрытый текст на отрезки, начало и конец которых выде-лить вставкой *контрольных меток* (специальных маркеров). В результате этого пропущенный при передаче символ приводит к неверному расшифро-ванию лишь до тех пор, пока не будет принят один из маркеров.

***Синхронные потоковые шифры уязвимы к атакам на основе изме-нения отдельных бит шифртекста***,**когда злоумышленник может изме-нить эти биты так, что шифртекст расшифруется так, как это ему вы-годно**.

Сформулируем в качестве итога краткого описания СПШ следующие их свойства:

* требования по синхронизации:

при использовании СПШ получатель и отправитель должны быть синхрони-зированы, т.е. должны вырабатывать одинаковые значения ключевого потока для соответствующих символов передаваемого потока данных. Если синхро-низация 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 рас-шифровании других символов шифртекста;

* свойство активной атаки:

любая вставка или удаление символа в шифртекст активным противником

приводит к нарушению синхронизации и обнаруживается получателем, рас-

шифровывающим 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 ведом-ствами, и в литературе почти не освещается.

84

* + **самосинхронизирующихся потоковых шифрах символы ключе-вой гаммы зависят от исходного секретного ключа шифра и от конечно-го числа последних знаков зашифрованного текста**.

Основная идея заключается в том, что внутреннее состояние генера-тора потока ключей является функцией предыдущих *s* битов шифртекста. Поэтому генератор потока ключей на приемной стороне, приняв *s* битов, ав-томатически синхронизируется с шифрующим генератором.

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

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

Таким образом, основные свойства ССПШ можно сформулировать следующим образом:

* *самосинхронизация*:

должна наступать при удалении или вставке некоторых символов шифртек-ста, поскольку процесс расшифрования зависит от некоторого фиксирован-ного числа предшествующих символов шифртекста. Это означает, что в слу-чае удаления некоторого символа из шифртекста сначала будут выступать ошибки при расшифровании, однако затем установится автоматически тре-буемый 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

* большей (по сравнению с синхронными шифрами) вероятностью будет за-мечено 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, т.к. там наступает рассинхронизация). Поэтому не-обходимы дополнительные механизмы для контроля этой ситуации;

* рассеивание статистики открытого текста:*

поскольку каждый символ открытого текста влияет на весь последующий шифртекст, статистические свойства открытого текста (далеко не случаен) не сохраняются в шифртексте.

85

***6.4 Генераторы псевдослучайных последовательностей для потоковых шифров***

***6.4.1 Линейные конгруэнтные генераторы***

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

*Требования к криптографически стойкому генератору ПСП* можноразделить 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.

Смысл указанного теста состоит в следующем*:* *не должно существо-вать полиномиального алгоритма, который, на основе первых k бит случай-ной последовательности, «сможет» предсказать (k+1)-й бит с вероятно-стью более 0,5.*

Относительно второго требования. Генератор должен оставаться на-дежным даже в случае, когда часть или все его состояния стали известны (или были вычислены).

Наиболее часто используемый алгоритм генерирования (программно или аппаратно) ПСП реализуется на основе так называемого *линейного кон-груэнтного генератора*,описываемого следующим рекуррентным соотноше-нием:

|  |  |
| --- | --- |
| *xt+1 = (a\*xt + c) mod n,* | (6.3) |

где: *xt* и *xt+1* - соответственно *t*-й (предыдущий) и (*t+1*)-й (текущий, вычис-ляемый) члены числовой последовательности; *а*, *с* и *n* – константы. Период такого генератора не превышает *n*.

При *с* = 0 получаем *мультипликативный конгруэнтный генератор* ПСП. *Примеры параметров* генератора на основе (6.3) для РС с 32-разрядной архитектурой: *n*= 231 -1 = 2 147 483 647, *а*=16807; 630360016; 10783183814; 1203248318; 397204094.

86

* сравнении с генераторами на основе (6.3) обеспечивают более вы-сокую криптостойкость подобные блоки, описываемые следующими соот-

ношениями:

*xt+1* = (1176\**xt* + 1476\**xt-1* + 1776\**xt-2*) mod 232- 5, *xt+1* = (213(*xt + xt -1 + xt-2*) mod 232- 5,

*xt+1* = (1995\**xt* + 1998\**xt-1* + 2001\**xt-2*) mod 232-849, (6.4)

*xt+1* = (219(*xt + xt-1 + xt-2*) mod 232–1629.

Особенностями генератора, задаваемого (6.3) и (6.4), является то, что начальное значение генератора, *х0*, является тайной; кроме того, если извест-но текущее (или предыдущие) значение ПСП, а также используемые кон-станты, то не представляет труда криптоаналитику вычислить всю последо-вательность. В силу последнего обстоятельства данный тип генераторов не находит практического применения в серьезных ИС.

Комбинации нескольких (чаще двух) линейных конгруэнтных генера-торов позволяют значительно повысить период ПСП. Б. Шнайер19, например, приводит данные о том, как на 32-разрядных ПК реализовать генератор в ви-де комбинации двух, каждый из которых обеспечивает период соответствен-но 231 – 85 и 231 – 249, а комбинированный генератор позволяет достичь пе-риода 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 си-лу простоты реализации на основе цифровой логики. РС с линейной обрат-ной связью (РСЛОС) состоит из двух частей: собственно РС и *функции об-ратной связи*.На рис. 6.2представлена общая схема РС с линейной обратнойсвязью. Функция обратной связи реализуется с помощью сумматоров сложе-ния по модулю два (элементы XOR; на рис. 6.2 обозначены в виде кружочков со знаком сложения).

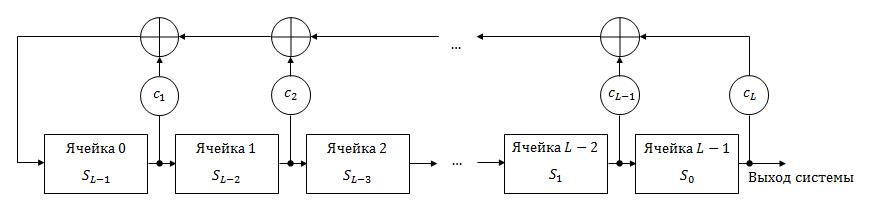


Рисунок 6.2. Общая схема регистра сдвига с линейной обратной связью

1. см. Б. Шнайер. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.:Изд-во ТРИУМ, 2003

87

Последний разряд РС в каждом такте формирует очередной символ ПСП. которые называют также *М-последовательностями* (или последова-тельностями максимальной длины).

Количество ячеек *L*, называют длиной регистра. Биты (ячейки) *i* обычно нумеруются числами от 0 до *L*-1. содержимое *i*-й ячейки обозначает-ся через *SL-1-i*.

* течение каждой единицы времени (за такт) выполняются следую-щие операции:

содержимое ячейки *L*-1 формирует часть выходной последовательно-

сти;

содержимое *i*-й ячейки перемещается в (*i+*1)-ю ячейку, новое содер-жимое ячейки *0* определяется битом обратной связи, который вычисляется сложением по модулю с определѐнными коэффициентами *ci* битов ячеек.

Помимо этого, выходная последовательность зависит, главным обра-зом, от начального состояния каждой ячейки регистра.

При аппаратной реализации РСЛОС отдельная ячейка может пред-ставлять собой триггер. В каждом новом такте состояние *i*-го триггера (его выходное состояние) передается на вход (*i+*1)-го триггера и становится со-стоянием этого триггера.

РСЛОС строятся на основе *примитивных порождающих полиномов*,

которые мы подробно анализировали при изучении циклических помехо-устойчивых кодов. В частности, для схемы на рис. 6.2 такой полином *или ас-социированный многочлен* можно представить в следующем виде:

|  |  |
| --- | --- |
| *C(x) = 1+c1x+c2x2+ … + cLxL.* | (6.5) |

Его ненулевые коэффициенты *ci* называются *отводами*, (как и соот-ветствующие ячейки регистра, cоставляющие значения аргументов функции обратной связи). В этом соотношении первый член суммы (1) означает, сто результат вычислений в цепи обратной связи подается на вход 0-й ячейки.

Вспомним, что важным свойством многочлена *C(x)* является приво-димость. Многочлен называется *приводимым*, если он может быть представ-лен как произведение двух многочленов меньших степеней с коэффициента-ми из данного поля (в нашем случае с двоичными коэффициентами). Если нет, то многочлен называется *неприводимым*. Если многочлен является не-приводимым, 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.

Рассмотрим пример.

*Пример 6-1*.Пусть задан многочлен,задающий структуру РСЛОС:

*C(x) = 1+c1x+ c3x3 = x3 + x + 1***;**

как видим, для нашего случая коэффициенты полиномы равны следующим значениям: *c3= 1, c2* *= 0, c1= 1* .

88

* соответствии со значениями коэффициентов полиномы принято для краткости обозначать числами, соответствующими степеням ненулевых сла-гаемых полинома. Таким образом, для нашего случая это будет ряд: 310. На рис. 6.3 представлена структурная схема, соответствующая заданному поли-ному.

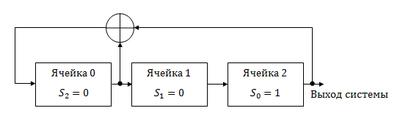


Рисунок 6.3. Структурная схема РСЛОС, заданная полиномом *C(x) = 1+c1x+ c3x3*

Положим, что начальное состояние ячеек регистра будет *001*.

После первого такта работы генератора будут выполнены следующие операции:

содержимое ячейки 2 сформирует первый символ выходной последо-вательности (ПСП) генератора: этим символом будет *1*;

* ячейку 0 будет записана сумма по модулю два начальных значений 0-й и 2-й ячеек: (*0+1*) *mod 2* = 1;
* 1-ю ячейку будет записано состояние 0-й ячейки: *0;*

во 2-ю ячейку будет записано состояние 1-й ячейки: *0.*

Таким образом, после первого такта (цикла) работы генератора со-стояние 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+c1x+ c3x3* (310)

является неприводимым.

Другими примерами неприводимых полиномов являются: 210, 320, 310, 410, 520, ….., 84320, … .

*Вопросы для контроля и самоконтроля*

1. Пояснить особенности потоковых шифров.
2. Дать классификацию потоковых шифров.
3. Пояснить достоинства и недостатки синхронных и самосинхрони-зирующихся потоковых шифров.
4. Нарисовать общую структурную схему генератора ПСП на основе

РЛОС.

89

1. Записать в виде полинома следующую упрощенную форму его представления: 410, 520, 654321, 6420, 84320.
2. Представить структурную схему регистра сдвига по заданию 5. Составить таблицу его состояний и определить период генерируемой ПСП.

90

**ГЛАВА III** **СТЕГАНОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ**

1. **Структура, особенности построения и использования стеганографических систем**

***7.1 Терминология, сущность и цели стеганографического***

***преобразования информации***

*Стеганография* –это,в общем случае,искусство передачи скрытогосообщения. Принято считать, что впервые этот термин использовал Трите-миус (1462–1516) в своей работе под тем же названием: «Стеганография».

Слово «стеганография» в переводе с греческого буквально означает тайнопись (от греч. στεγανός – скрытый + γράυω – пишу; буквально «тайно-пись»).

Местом зарождения стеганографии многие называют Египет, хотя пер-выми «стеганографическими сообщениями» можно назвать и наскальные ри-сунки древних людей. Первое упоминание о стеганографических методах в литературе приписывается Геродоту, который описал случай передачи сооб-щения 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 посла-ния на голове раба. Для передачи тайного сообщения голову раба обривали, наносили на кожу татуировку, и, когда волосы отрастали, отправляли

* посланием. В Китае письма писали на полосках щелка. Поэтому для сокры-тия сообщений полоски с текстом письма сворачивались в шарики, покрыва-лись воском и затем глотались посыльными. Темное средневековье породило не только инквизицию: усиление слежки привело к развитию как криптогра-фии, так и стеганографии. 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 пе-реписки. В штат «черных кабинетов», помимо криптографов и дешифро-вальщиков, 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 ис-торический эпизод: восставшими дворянами в Бордо был арестован франци-сканский монах Берто, являвшийся агентом кардинала Мазарини. Восстав-шие разрешили Берто написать письмо знакомому священнику в город Блэй. Однако в конце этого письма религиозного содержания монах сделал при-писку, 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 результа-те монах Берто был спасен.

91

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 и в годы гражданской войны между южана-ми и северянами. Так, в 1779 году два агента северян Сэмюэль Вудхулл и Ро-берт Тоунсенд передавали информацию Джорджу Вашингтону, используя специальные чернила.

Различные симпатические чернила использовали и русские револю-ционеры в начале XX века. Симпатические чернила или, с успехом выпол-няющее их роль, обычное молоко – один из самых распространенных стега-нографических методов. Многие книги Ленина были написаны в местах за-ключения 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 пар-тии. Впрочем, царская охрана тоже знала об этом методе (в ее архиве хранит-ся документ, в котором описан способ использования симпатических чернил

* приведен текст перехваченного тайного сообщения революционеров).
  + середине 20-го 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, обла-дали способностью передавать большие объемы информации, включая чертежи
* рисунки.

Сам метод был придуман намного раньше, почти сразу после изобрете-ния принципа фотографической печати. Микроточки появились сразу же по-сле изобретения 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 этот метод претерпел второе рождение и успех его был весьма заметным. Американцы, впечатленные достижениями своего противника в стеганографии, после войны запретили даже такие отно-сительно 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 сообщений20.

Таким образом, *стеганография* *–* *это наука о способах передачи* *(хра-нения) сокрытой информации, при которых скрытый канал организуется на базе и внутри открытого канала с использованием особенностей воспри-*

1. Все о хакерах, хакинге и защите информации: компьютерная стеганография [Электронный ресурс]. – Режим дос-

тупа: [http://supermegayo.ru/ compterr/77.html.](http://supermegayo.ru/%20compterr/77.html) – Дата доступа: 16.12.2011

92

*ятия информации*,причем для этой цели могут использоваться такие прие-мы, как:

– полное сокрытие факта существования скрытого канала связи;

– создание трудностей для обнаружения, извлечения или модифика-ции передаваемых сокрытых сообщений внутри открытых сообщений-контейнеров;

– маскировки сокрытой информации в протоколе.

Компьютерная стеганография изучает способы сокрытия информации

* компьютерных данных, представляющих собой различные файлы, про-граммы, пакеты протоколов и т. п.

Современная стеганография, как правило, имеет дело с электронны-ми средствами. Компьютерная стеганография базируется на двух принципах. Первый заключается в том, что файлы, содержащие оцифрованное изобра-жение или звук, могут быть до некоторой степени видоизменены без потери функциональности, в отличие от других типов данных, требующих абсолют-ной точности.

Второй принцип состоит в неспособности органов чувств человека раз-личить незначительные изменения в цвете изображения или качестве звука, что особенно легко использовать применительно к объекту, несущему избы-точную 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 пиксела, не приводит к сколь-нибудь заметному для человека изменению цвета.

Особенностью стеганографического подхода является то, что он не предусматривает прямого оглашения факта существования защищаемой ин-формации. Это обстоятельство позволяет в рамках традиционно существую-щих информационных потоков или информационной среды решать некото-рые важные задачи защиты информации ряда прикладных областей.

Основным определяющим моментом в стеганографии является *стега-нографическое преобразование*.До недавнего времени стеганография,какнаука, в основном изучала отдельные методы сокрытия информации и спо-собы их технической реализации. Разнообразие принципов, заложенных в стеганографических методах, по существу тормозило развитие стеганогра-фии 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 ка-чественных и количественных оценок стеганометодов. В этом история разви-тия стеганографии резко отличается от развития криптографии.

Как и любое новое направление, компьютерная стеганография, несмот-ря на большое количество открытых публикаций и ежегодные конференции, долго не имела единой терминологии. На конференции Information Hiding:

93

First Information Workshop в 1996 году было предложено использовать единую терминологию и обговорены основные термины21.

**Определение 7.1***.* ***Стеганографическая система* (стегосистема22) – совокупность средств и методов, которые используются для формиро-вания скрытого канала передачи информации**.

Стегосистема образует стегоканал, по которому передается (или в котором хранится) заполненный контейнер. Этот канал считается подвер-женным воздействиям со стороны нарушителей.

При построении стегосистемы должны учитываться следующие по-ложения:

– противник имеет полное представление о стеганографической сис-теме и деталях ее реализации; единственной информацией, которая остается неизвестной потенциальному противнику, является ключ, с помощью кото-рого 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 сооб-щения в других данных до тех пор, пока ключ хранится в тайне;

– потенциальный противник должен быть лишен каких-либо техниче-ских и иных преимуществ в распознавании или раскрытии содержания тайных сообщений.

Любая стегосистема должна отвечать следующим требованиям:

– *свойства контейнера должны быть модифицированы*, чтобы изме-

нение невозможно было выявить при визуальном контроле. Это требование определяет качество сокрытия внедряемого сообщения: для обеспечения беспрепятственного прохождения стегосообщения по каналу связи оно нико-им 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 по-терей данных;

– для *сохранения целостности встраиваемого сообщения* необходи-

мо 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*.

1. Pfitzmann, B. Information Hiding Terminology, in Information Hiding / B. Pfitzmann // Springer Lecture Notes in Computer Science. – v.1174. – 1996. – P. 347–350
2. в литературе встречаются два сокращенных термина, обозначающие саму стеганографическую систему или ее элементы: например, ***стего****система* и ***стегано****система*

94

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 сис-теме является ни чем иным как носителем сокрытой информации, поэтому вполне возможно использование и такого термина. В некоторых источниках термин контейнер также заменяют названием «стего», который также являет-ся производным от английского сокращения «stego» (полное название «stega-no»).

**Определение 7.2 *Контейнером* (носителем) *С* называют несекрет-ные данные, которые используют для сокрытия сообщений** (рисунок7.1).

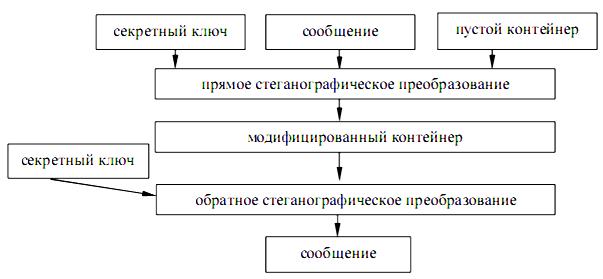


Рисунок 7.1 Общая схема стеганографического преобразования информации

*Пустой контейнер* (немодифицированный контейнер) –это некоторый кон-тейнер *С*, не содержащий сообщения. *Заполненный контейнер* (модифициро-ванный контейнер) – это элемент *С* системы, содержащий сообщение *М*.

* компьютерной стеганографии в качестве контейнеров могут быть ис-пользованы различные оцифрованные данные: растровые графические изо-бражения, цифровой звук, цифровое видео, всевозможные носители цифро-вой информации, а также текстовые и другие электронные документы.

Выделяют два типа контейнера: *потоковый* и *фиксированный*.

Потоковый контейнер представляет собой последовательность битов, ко-торая 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 скры-тое сообщение. Сообщение встраивается в контейнер в реальном масштабе вре-мени, поэтому в кодере заранее неизвестно, хватит ли размеров контейнера для

95

передачи всего сообщения. В то же время, в один контейнер большого объема может быть встроено несколько сообщений.

* + фиксированном контейнере известны его размеры и характеристики. Это позволяет выполнять встраивание данных оптимальным способом.

*Контейнер может быть избранным, случайным или навязанным*.Избран-

ный контейнер зависит от встроенного сообщения, а в предельном случае явля-ется его функцией. Такой тип контейнера характерен именно для стеганографии. Навязанный контейнер появляется, когда тот, кто предоставляет контейнер, по-дозревает о возможной скрытой переписке и желает предотвратить ее. На прак-тике чаще всего имеют дело со случайным контейнером.

Скрытие информации большого объема выдвигает существенные требова-ния 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 необ-ходимо преобразовать в определенный удобный для упаковки вид. Кроме того, для повышения защищенности секретной информации последнюю можно за-шифровать достаточно устойчивым криптографическим ключом.

**Определение 7.3. *Сообщением* *(стегосообщением)* *М* называют секрет-**

**ную информацию, наличие которой в контейнере необходимо сокрыть**.

**Определение 7.4. *Ключом* *(стегоключом)* *К* называют секретную информацию, известную только законному пользователю, которая опре-деляет конкретный вид алгоритма сокрытия**.

Через *К* обозначается множество всех допустимых секретных ключей. Заметим, что в работах по стеганографии ключ понимается как в широком, так

* в узком смысле. В широком смысле стеганографический ключ – это сам не-

известный противнику способ сокрытия информации.

При этом необходимо отличать стеганографический ключ от криптографического, который также может присутствовать в системе

* использоваться для предварительного криптографического закрытия вне-дряемой информации.

По аналогии с криптографией, по типу стегоключа *стеганографиче-*

*ские системы можно подразделить на три типа*:

– *с секретным ключом;*

– *с открытым ключом;*

– *смешанные*.

* стегосистеме с секретным ключом используется один ключ, который должен быть определен либо 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 авторизо-ванными лицами. Отправитель, встраивая секретное сообщение в избранный контейнер *С*, использует стегоключ *K*. Если получатель знает данный ключ,

96

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 извлече-ния скрытой информации дополнительно необходимы сведения о первичном (пустом) контейнере или некоторые другие данные, отсутствующие в стега-нограмме. Такие системы представляют ограниченный интерес, поскольку они требуют передачи изначального вида контейнера, что эквивалентно тра-диционной задаче обмена ключами.

Для функционирования стеганографической системы с открытым клю-чом необходимо иметь два стегоключа: один секретный, который необходи-мо хранить в тайне. А другой – открытый, который может храниться в дос-тупном для всех месте и может передаваться свободно по незащищенному каналу связи. Ключи различаются таким образом, что с помощью вычисле-ний 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, кроме алгоритма стеганографического преобразования, отсутствует необходимость в дополни-тельных данных, наподобие стегоключа.

На практике преимущество отдается именно бесключевым стегосисте-мам, хотя они могут быть раскрыты в случае, если нарушитель узнает о ме-тоде стеганопреобразования, который был при этом использован. В связи с этим в бесключевых системах часто используют особенности криптографи-ческих систем с открытым и/или секретным ключом.

***7.2 Основные направления использования***

***стеганографическихсистем***

Современная стеганография, как правило, «имеет дело» с электронны-ми 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 элек-тронные контейнеры гораздо проще скрывать данные и извлекать их. Во-вторых, процедура осаждения/извлечения может быть автоматизирована с помощью специальных программных средств. В-третьих, электронный фор-мат данных характеризуется информационной избыточностью, которой мож-но управлять, чтобы скрыть сообщения. Эти общие для всех электронных до-кументов особенности, на наш взгляд, ставят знак равенства между цифровой и компьютерной стеганографией.

На рисунке 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.

97

Использование стеганографических систем является наиболее эффек-тивным при решении *проблемы защиты информации с ограниченным досту-пом*.Это означает возможность скрытой передачи информации.

Кроме указанного направления, **стеганография является одним** **из перспективных средств *для аутентификации и маркировки автор-ской продукции* с целью защиты авторских прав на цифровые объектыот пиратского копирования**.

На компьютерные графические изображения, аудио продукцию, ли-тературные произведения (программы в том числе) наносится специальная метка, которая остается невидимой для глаз, но распознается специальным программным обеспечением.

Методы компьютерной стеганографии



Признаки классификации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | Способ выбора | | |  |  |  | Доступ |  | Тип организации | | | | |  |  |
|  |  |  | контейнера | |  |  | к информации | |  |  | контейнера | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | | |  |  |  | |  |  |  |  |  | |  |
|  |  |  | Суррогатные | | |  |  | Потоковые | |  |  |  | Систематические | | |  |
|  |  |  |  |  |  |
|  |  |  | Селективные | | |  |  | С произвольным | |  |  |  | Несистематические | | |  |
|  |  |  |  |  |  |
|  |  |  | Конструирующие | | |  |  | доступом | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Тип информацион- | | |  |  |  | Используемые | | |  |  |  | Предназначение | | |  |  |
|  |  | ной среды |  |  |  |  | принципы | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | |  |  |  |  |  | |  |
|  |  | Текстовая |  |  |  |  | На основе избы- | | |  |  |  | Защита передаваемых или | | |  |
|  |  |  |  |  |  |  |
|  |  | Звуковая |  |  |  |  | точности среды | | |  |  |  | хранящихся данных | | |  |
|  |  |  |  |  |
|  |  | Стоп-кадр/видео |  |  |  |  | Структурные ме- | | |  |  |  | Защита прав на цифровые | | |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | тоды | | |  |  |  | объекты | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



Рисунок 7.2 Общая классификация методов компьютерной

стеганографии

*Метка содержит скрытую информацию, подтверждающую авторст-во. Скрытая информация призвана обеспечить защиту прав интеллектуаль-ной собственности*.

* качестве внедряемой информации можно использовать данные об ав-торе, дату и место создания произведения, номера документов, подтвер-ждающих авторство, дату приоритета и т. д. В качестве иллюстрации более подробно рассмотрим использование стеганографии для защиты авторских прав на текстовые документы.

98

***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 функ-циональности, то текстовые документы, коды программ или базы данных требуют абсолютной точности при обратных преобразованиях. Это обстоя-тельство чрезвычайно важно, если, например, текстовый документ-контейнер

* осажденной информацией претерпевает конвертацию на основе иного сти-ля (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 сообще-ния, и *лингвистические* методы, которые основаны на эквивалентной транс-формации текстовых файлов, сохраняющей смысловое содержание текста, его семантику.

***7.2.1.1 Синтаксические методы текстовой стеганографии***

К числу этих методов этого подкласса относят следующие.

***Line-shift coding*** (изменение расстояния между строками электронно-го текста). Он также называется *методом изменения межстрочных интерва-лов*.Его сущность заключается в том,что используется текст с различнымимежстрочными расстояниями. Выделяется максимальное и минимальное рас-стояния между строками, позволяющее кодировать соответственно символы «1» и «0» осаждаемого сообщения (рисунок 7.3). Разница в межстрочных расстояниях авторами изменялась на 1/300 дюйма (это расстояние было при-вязано 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 кон-тейнере.

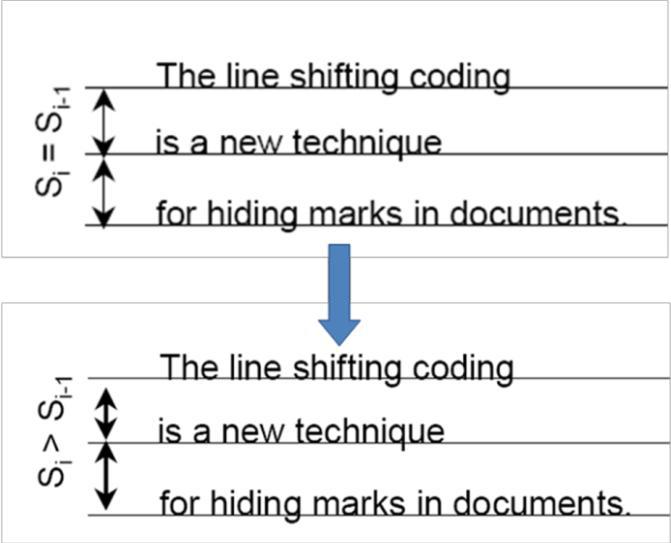


Рисунок 7.3 — Иллюстрация метода на основе изменения межстрочных интервалов

99

***Word-shift coding*** (изменение расстояния между словами в однойстроке электронного текста). Суть метода состоит в том, что осаждение ин-формации 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, увеличи-вают или уменьшают до размеров уже выделенных. Частный случай этого метода — *метод изменения количества пробелов* (рисунок 7.4). Данный ри-сунок показывает пример внедрения в текст-контейнер бинарной последова-тельности 0101100100111010. Как видно, переход с одинарного пробела на двойной кодирует «1» (пары выделены красным фоном), переход же с двой-ного пробела на одинарный кодирует «0» (пары выделены синим фоном).

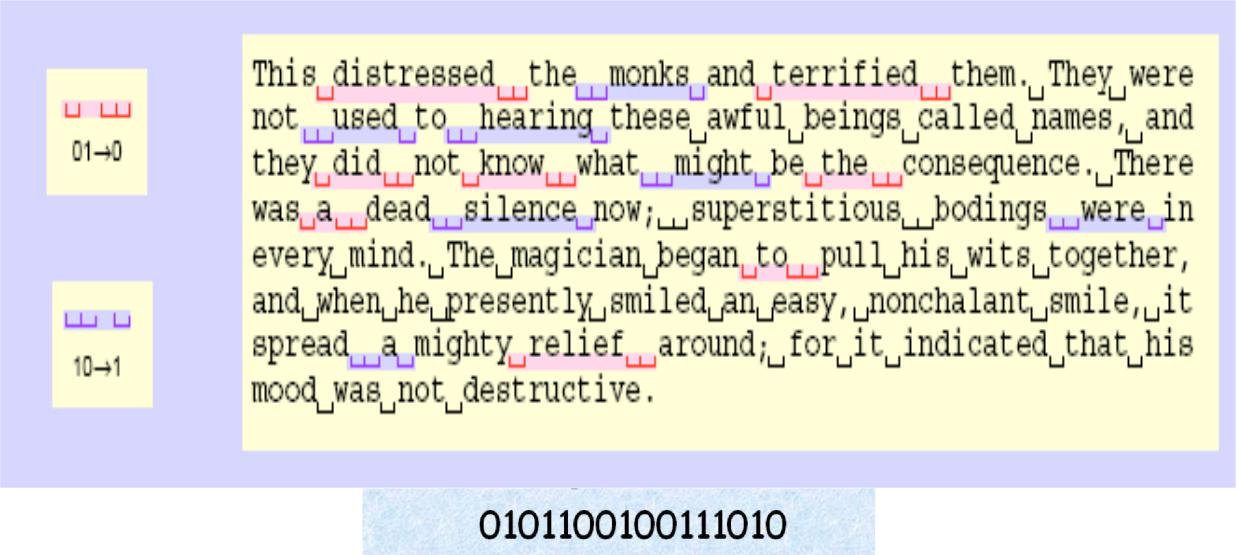
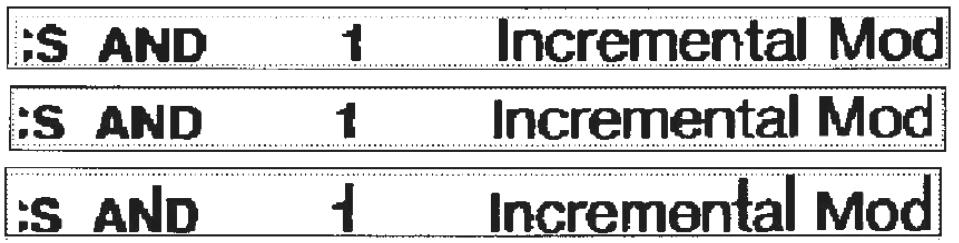


Рисунок 7.4 — Пример использования метода на основе длин пробелов

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, на-пример белые стихи, не имеют четких признаков конца). В-третьих, тексто-вые редакторы часто автоматически добавляют символы пробела после точ-ки.

***Feature coding*** (внесение специфических изменений в шрифты(на-чертания отдельных букв). Этот метод заключается в изменении написания отдельных букв используемого стандартного шрифта (рисунок 7.5).



100

Рисунок 7.5 — Иллюстрация метода feature coding

Визуально заметны различные образы, соответствующие буквам с верхними (напр., *l*, *t*, *d*) или нижними (напр., *a*, *g*) выносными элементами. Например, букву «А» можно модифицировать, незначительно укорачивая длинную нижнюю часть буквы. При этом можно закодировать стегосообще-ние так, что модифицированная буква будет означать «1», а немодифициро-ванная — «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 рисунке 7.6.



а) пустой контейнер; б) заполненный контейнер (со стегосообщением

«1»)

Рисунок 7.6 — Пример применения метода feature coding

Известны иные методы текстовой стеганографии. Кратко охарактери-зуем их.

***Метод изменения интервала табуляции***.Аналогичен вышеописан-ному методу изменения количества пробелов, только в этом случае меняется не количество пробелов, а соответственно расстояние между строками и ин-тервал табуляции;

***Null chipper*** (дословно—несуществующий,нулевой лепет).Предпо-лагает размещение тайной информации на установленных позициях слов или в определенных словах текста-контейнера, который, как правило, ли-шен логического смысла (как видно, действительно лепет). На рисунке 7.7 показан пример реализации метода — скрытой информацией являются пер-вые символы слов.

101

«President's embargo ruling should have immediate notice. grave sit-uation affecting international law. statement foreshadows ruin of many neutrals. yellow journals unifying national excitement immense-ly».



Pershingsailsfromnyjunei **** Pershing Sails From NY June I.

Рисунок 7.7 — Пример реализации метода null chipper

***Метод увеличения длины строки***.Предусматривает искусственноеувеличение длины каждой строки за счет пробелов: например, одному про-белу соответствует логический 0, двум — 1 (рисунок 7.8).

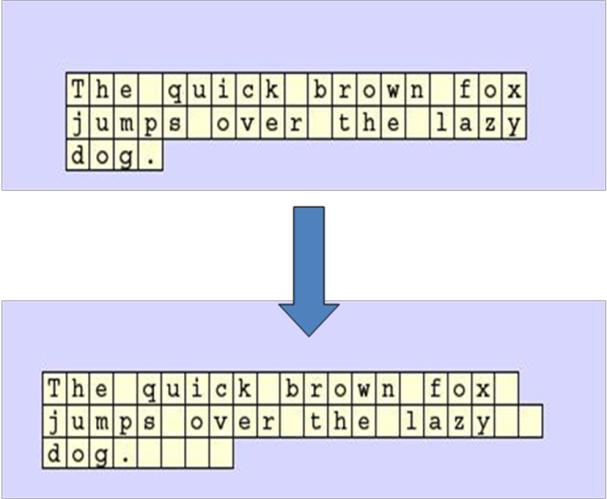


Рисунок 7.8 — Иллюстрация метода увеличения длины строки

Преимущество такого метода кодирования состоит в том, что оно мо-жет быть выполнено с любым текстом; изменения в формате резко не броса-ются 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 со-общения, представленного единицей, используется символ нижнего регистра,

* нулем — верхнего (или наоборот). Например, секретный текст, состоящий из одной буквы «А», необходимо внедрить в текст-контейнер «steganography». Для этого используем двоичное представление кода символа «А» — «01000001». Далее предположим, что для обозначения бита секретно-

102

го сообщения, представленного единицей, используется символ верхнего ре-гистра, а нулем — нижнего.

Результат внедрения секретного сообщения «А» в текст-контейнер «steganography» показан на рисунке 7.9.

«А» — «01000001»

Контейнер — «steganography»

«А» + «steganography» → «01000001» + «steganogra-

phy» → «sTeganoGraphy»

Рисунок 7.9 — Иллюстрация метода на основе регистра клавиатуры

Существует модификация данного метода, основанная на применении различных алфавитов, в которых используются символы, имеющие одинако-вое начертание, но различную кодировку (например, *а*, *в*, *е*, *р*, *т*, с).

***Метод невидимых символов***.Пробел кодируется символом с ко-дом 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 исход-ный 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 стеганографические методы, которые интерпретируют текст как двоичное изображение. Необходимо от-метить, что данные *методы нечувствительны к изменению масштаба доку-мента, что обеспечивает им хорошую устойчивость к большинству иска-жений, которые могут иметь место при активных атаках*.

Описанные выше методы — синтаксические — легко применяются

* любому тексту, независимо от его содержания, назначения и языка. *Син-*

*таксические системы стеганографии легко реализуются в программном ко-*

*де*,так как они полностью автоматические и не требуют вмешательства опе-ратора. Однако синтаксические методы неустойчивы к форматированию тек-ста (вспомним робастность систем на основе ЦВЗ), и поэтому информация может быть потеряна при простом применении иного стиля форматирования текста-контейнера, скрывающего в себе стегосообщение. К тому же с помо-щью синтаксических методов можно передать незначительное количество информации. В литературе отсутствуют результаты экспериментального ис-следования эффективности проанализированных методов с проверкой на больших объемах данных.

103

***7.2.1.2 Лингвистические методы текстовой стеганографии***

Стеганографические методы, основанные на лексической структуре текста, обладают большими возможностями.

***Лингвистическая стеганография занимается скрытым кодировани-ем произвольной информации, представленной в двоичном виде, в тек-стах***.

Необходимо отметить, что осмысленность и внешняя «безобидность» текста должна сохраниться. К *лексическим* методам встраивания скрытой информации в текстовые файлы-контейнеры относят *метод переменной дли-ны слова, метод первой буквы, метод синонимов* и другие.

Одним из наиболее обсуждаемых методов является ***метод*,** **основан-ный на *системе синонимов языка***,используемого для написания электрон-ного текста. Проведенные исследования для случая английского языка, пока-зали, что среднее количество синонимов в одном подмножестве синонимов равняется 2,56. Минимальное количество синонимов в одном множестве си-нонимов равняется 2, а максимальное 13.

*Пример 7-1*.В качестве примера приведем множество синонимовS0:{«propensity», «predilection», «penchant», «proximity»}. В приведенном мно-жестве синонимов, каждое слово имеет единственное одинаковое смысловое значение, что позволяет закодировать каждое слово своим уникальным ко-дом, например, «propensity» - 00, «predilection» - 01, «penchant» - 10, «proximi-ty» - 11. Подобное кодирование позволяет выбирать одно из четырех слов в зависимости от двух бит секретного сообщения. Отметим, что при этом, не-зависимо какое из четырех слов будет выбрано, семантика (смысл) сообще-ния не изменится.

Процедура передачи секретного сообщения, используя лексическую стеганографию, 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 последо-вательность: ...01000... , используя по возможности криптографические мето-ды.
  3. Отправитель, последовательно анализируя текстовый файл, находит первое слово, для которого существуют *N* синонимов.
  4. Отправитель вычисляет целую часть значения *log2* *N,* которая опре-деляет число символов секретного сообщения, которые могут быть внедрены
* контейнер путем выбора соответствующего синонима. Например, если в тексте встретилось слово «penchant», принадлежащее множеству *S0:* {«pro-pensity», «predilection», «penchant», «proximity»}, в соответствии со значени-

ем двух бит закодированной секретной информации - 01, это слово должно быть заменено словом синонимом «predilection».

104

Аналогичные действия выполняет и получатель. Получатель анализи-рует 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 *log2N* определяет число бит, которые закодированы на основании текущего множе-ства синонимов. Например, если получатель обнаружит в тексте слово "predi-lection" и определит, что оно относится к множеству синонимов *S0* состояще-му из *N*=4 синонимов, тогда *log2* *N=2* бита, а слово "predilection" интерпрети-руется, как два бита (01) секретного сообщения. Следует отметить, что в ка-ждом подмножестве синонимов их упорядочивание должно выполняться по одному и тому же алгоритму и у отправителя сообщения и у его получателя. Например, в алфавитном порядке.

* + случае слов с несколькими смысловыми значениями подобное коди-рование оказывается невозможным. Также невозможно кодирование, если один из синонимов состоит из двух (или более), разделенных пробелом, слов.

К сожалению, количество синонимов не всегда равно 2k , где *к* - целое положительное число. Например, имеем три подмножества синонимов S0: {AAA, ВВВ, ССС}; S1{МММ, NNN, ООО, РРР, QQQ}; S2:{WWW, XXX, YYY}.

Использование данных подмножеств синонимов согласно приведенной ранее идеи 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 зако-дировать 3\*5\*3=45 состояний или не менее, чем |log2 45|=5 бит.

Для увеличения объема внедряемой в контейнер секретной информа-ции может быть использована система счисления со смешанным основанием.

* 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 log2 45 = 5 бит (берем, понятно, целочисленное значение) вместо 4. Для этого использу-ем 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 переда-ваться или храниться тайная информация**.

Так, при использовании ***метода переменной длины*** пользователю, ко-торый хочет послать секретное сообщение, необходимо сгенерировать (на-брать) текст, в котором слова должны иметь соответствующую длину. Длина этих слов зависит от секретного сообщения и способа кодирования. Обычно одно слово текста-контейнера определенной длины кодирует два бита ин-формации из стегосообщения. Например, слова текста длиной в 4 и 8 симво-

105

лов 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 би-та. 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 не содержащим смысла.

Другим, не менее распространенным лексическим методом передачи скрытой информации является ***мимикрия.*** Мимикрия генерирует осмыслен-ный текст, используя синтаксис, описанный в ***Context Free Grammar*** (CFG),

* 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. Ми-микрия создает бинарное дерево, которое основано на возможностях CFG и составляет текст, выбирая те листья дерева, которые кодируют нужный бит.

*Пример 7-2*.Например,необходимо передать секретное сообщение10101, 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» на следующей диаграмме, приведенной в таблице 7.1.

Таблица 7.1 – Внедрение секретного кода на основе бинарного дерева

|  |  |  |  |
| --- | --- | --- | --- |
| Узлы принятия решения | Скрытый | Ответы, полученные в результате вне- |  |
| бит | дрения 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 | 1 | **что →** одежду |  |
| **что** |  |
|  |  |  |
|  |  | 106 |  |

Окончательно получилось следующее предложение: *Иван поехал до-мой, чтобы забрать одежду*.

Недостатками этого метода является то, что с его помощью нельзя пе-редавать большие объемы информации, а также низкая производительность метода. Кроме того, необходимо отметить невысокую скрытность секретного сообщения, которое в сильной мере влияет на структуру передаваемого тек-ста.

На наш взгляд, еще одним методом, основанным на мимикрии, явля-ется ***Spammimic*** (уподобление спаму). Здесь в качестве контейнера использу-ется обычный спам (или любой нейтральный текст, см. рисунок 7.10), внутри которого размещаются установленным обеими сторонами способом знача-щие символы (стегосообщение).

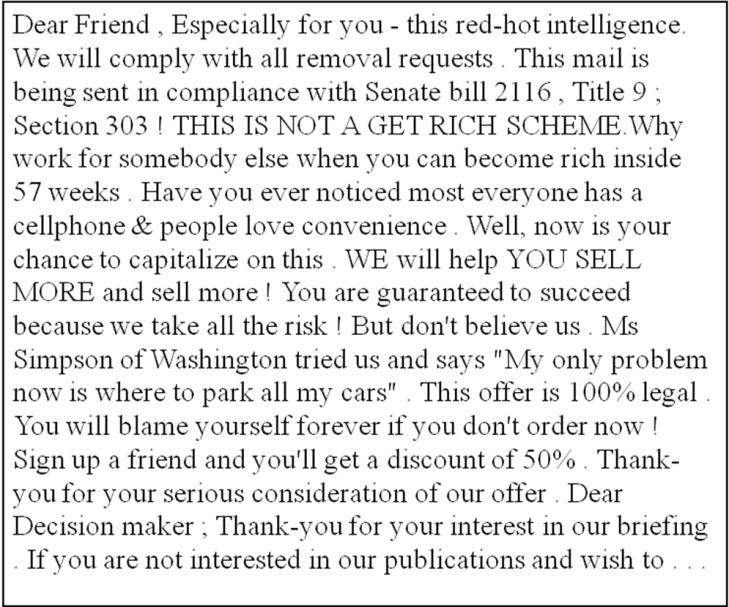


Рисунок 7.10 — Текст-контейнер с осажденным в нем сообщением «*Здравствуйте*!»

Существует и множество других методов преобразования текста. В любом случае, при разработке эффективных лексических стеганографиче-ских методов, необходимо искать золотую середину: контейнер должен быть «плотно» насыщен стегоинформацией и при этом совершенно не должен вы-деляться из обычной общей массы файлов такого же формата и наполнения.

* таблице 7.2 представлены результаты выполненного авторами данно-го пособия сравнительного анализа эффективности некоторых из проанали-зированных методов.

Таблица 7.2 — Сравнительные результаты анализа эффективности син-таксических методов текстовой стеганографии

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Стегозна- | Плотность | Часть стегосо- |

107

|  |  |  |  |
| --- | --- | --- | --- |
|  | ков | заполнения, | общения, содер- |
|  |  | % | жащаяся в каж- |
|  |  |  | дом 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 координат23 | 63 328 767 | 100,0 | 3,000 |

Под «плотностью заполнения» (третий столбец таблицы) будем пони-мать отношение стегознаков в пустом контейнере к общему числу символов

* пустом контейнере. Например, для метода изменения регистра буквы и ме-тода feature coding стегознаками являются буквы любого алфавита, т.е. про-белы, цифры, специальные знаки и символы не учитываются. А в методах изменения цвета символов, изменения масштаба символов стегознаками яв-ляются все символы документа, в том числе специальные знаки и символы.

*До сих пор вопрос о создании безопасной лингвистической стегоси-стемы остается открытым. Любая обработка текста редактором, его печать или перевод в другой формат может изменить расположение про-белов и уничтожить скрытый текст*.

***7.2.2 Стеганографические методы на основы избыточности среды***

Если текст-контейнер рассматривать на основе определенной цвето-вой модели (bitmap или RGB), то для решения наших задач могут быть адап-тированы известные методы графической стеганографии. Имеется в виду то, что младшие разряды цифровых отсчетов, формирующих изображение, со-держат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и дает воз-можность скрытия конфиденциальной информации.

* числу классических методов данного класса относится *LSB* (Least Significant Bit – наименее значащий бит) и различные его модификации.

***Метод LSB***.Основывается на ограниченных способностях зрения илислуха человека, вследствие чего людям тяжело различать незначительные ва-риации цвета или звука. Рассмотрим это на примере 24-битного растрового RGB-изображения. Каждая точка кодируется 3-мя байтами, каждый байт оп-ределяет интенсивность красного (Red), зеленого (Green) и синего (Blue) цве-тов. Совокупность интенсивностей цвета в каждом из 3-х каналов определяет оттенок пикселя.

1. на основе метода LSB

108

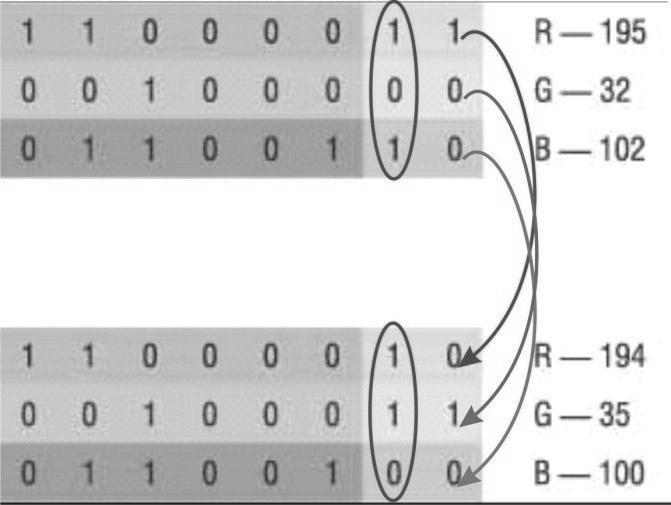


Рисунок 7.11 - Пример, показывающий принцип реализации метода LSB

Представим пиксель тремя байтами в битовом виде, как это показано на рисунке 7.11. Младшие биты (выделены бледным, справа) дают незначи-тельный 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 скрыть6бит–101100.Разделим их на 3 пары и заместим этими парами младшие биты в каждом ка-нале. Получили новый цвет, очень похожий на первоначальный (к сожале-нию, в черно-белом изображении это практически не заметно). Оценим эф-фективность такого метода: используя 2 бита на канал, мы сможем прятать три байта информации на 4 пикселя изображения. А это уже где-то 25% кар-тинки. Например, в мегабайтовом файле можно спрятать 250 Кбайт инфор-мации, причем для невооруженного глаза этот факт останется незаметен.

Недостатки метода:

1. скрытое сообщение легко разрушить, например, при сжатии или отобра-жении;
2. не обеспечена секретность встраивания информации. Точно известно ме-стоположение зашифрованной информации. Для преодоления этого недос-татка можно встраивать информацию не во все пиксели изображения, а лишь в некоторые из них, определяемые по псевдослучайному закону в соответст-вии с ключом, известному 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 вычис-лительно трудоемким, кроме того, его надо повторять для каждого отдельно взятого контейнера. Главным недостатком этого метода является то, что про-цесс моделирования может быть повторен нарушителем, возможно обла-дающим большим вычислительным ресурсом, создающим лучшие модели, что приведет к обнаружению скрытого сообщения.

**При использовании любого стеганографического метода на осно-ве избыточности среды существует возможность повысить степень на-**

109

**дежности скрытия, жертвуя при этом объемом скрываемых данных**.Объем встроенных данных и степень модификации контейнера изме-

няется 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 разныеметоды стеганографии. В дополнение к таблице 7.2 в таблице 7.3 приведены сравнительные характеристики стеганографических методов в более общем плане.

Таблица 7.3 Сравнительные характеристики некоторых стеганографи-ческих методов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Стеганографический | Краткая характеристика | Недостатки | Достоинств |  |
| метод | метода | метода | а метода |  |
| 1. Методы использования специальных свойств компьютерных форматов данных | | | |  |
| Методы использования ком- | Поля расширения имеются во | Низкая степень | Простота использо- |  |
| пьютерных форматов данных | многих мультимедийных форма- | скрытности (устойчи- | вания |  |
|  | тах, они заполняются нулевой | вости к взлому). |  |  |
|  | информацией и не учитываются | Передача небольших |  |  |
|  | программой | ограниченных объе- |  |  |
|  |  | мов информации. |  |  |
| Методы использования из- | Основаны на изменении положе- | Слабая производи- | Простота использо- |  |
| вестного смещения слов, | ния строк и расстановки слов в | тельность метода, пе- | вания. Имеется |  |
| предложений, абзацев | предложении, что обеспечивается | редача небольших | опубликованное |  |
|  | вставкой дополнительных пробе- | объемов информации. | программное обес- |  |
|  | лов между словами | Низкая степень | печение реализации |  |
| Методы выбора определен- | Акростих - частный случай этого | скрытности. | данного метода |  |
| ных позиций букв (нулевой | метода (например, начальные бу- |  |  |
| шифр) | квы каждой строки образуют со- |  |  |  |
|  | общение) |  |  |  |
| Методы использования спе- | Основаны на использовании спе- |  |  |  |
| циальных свойств полей | циальных "невидимых", скрытых |  |  |  |
| форматов, не отображаемых | полей для организации сносок и |  |  |  |
| на экране | ссылок (например, использование |  |  |  |
|  | черного шрифта на черном фоне) |  |  |  |
| Методы скрытия в неисполь- | Информация записывается в | Слабая производи- | Простота использо- |  |
| зуемых местах памяти на | обычно неиспользуемых местах | тельность метода, пе- | вания. |  |
| дисках | диска (например, в нулевой до- | редача небольших | Имеется опублико- |  |
|  | рожке) | объемов информации. |  |
|  | ванное программ- |  |
|  |  | Низкая степень | ное обеспечение |  |
|  |  | скрытности. | реализации данного |  |
|  |  |  | метода |  |
|  |  |  |  |  |
| Методы использования ими- | Метод основан на генерации тек- | 1. Слабая производи- | Результирующий |  |
| тирующих функций (mimic- | стов и является обобщением акро- | тельность метода, | текст не является |  |
| function) | стиха. Для тайного сообщения | передача небольших | подозрительным |  |
|  | генерируется осмысленный текст, | объемов информации | для систем монито- |  |
|  | скрывающий само сообщение | 2. Низкая степень | ринга сети |  |
|  |  |  |  |
|  |  | скрытности |  |  |
|  |  |  |  |  |
| Методы удаления идентифи- | Скрываемое сообщение шифрует- | Проблема скрытия | Простота реализа- |  |
| цирующего файл заголовка | ся и у результата удаляется иден- | решается только час- | ции. |  |
|  | тифицирующий заголо- | тично. | Многие средства |  |
|  | вок,оставляя только шифрованные | Необходимо заранее | (White Noise Storm, |  |
|  | данные. | передать часть ин- | S-Tools), обеспечи- |  |
|  | Получатель заранее знает о пере- | формации получателю | вают реализацию |  |
|  |  |  | 110 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | даче сообщения и имеет недос- |  | этого метода с PGP |
|  | тающий заголовок |  | шифроалгоритмом |
| 2. Методы использования избыточности аудио и визуальной информации | | |  |
| Методы использования из- | Младшие разряды цифровых от- | За счет введения до- | Возможность скры- |
| быточности цифровых фото- | счетов содержат очень мало по- | полнительной инфор- | той передачи боль- |
| графии, цифрового звука и | лезной информации. Их заполне- | мации искажаются | шого объема ин- |
| цифрового видео | ние дополнительной информацией | статистические харак- | формации. |
|  | практически не влияет на качество | теристики цифровых | Возможность защи- |
|  | восприятия, что и дает возмож- | потоков. | ты авторского пра- |
|  | ность скрытия конфиденциальной | Для снижения ком- | ва, скрытого изо- |
|  | информации | прометирующих при- | бражения товарной |
|  |  | знаков требуется кор- | марки, регистраци- |
|  |  | рекция статистиче- | онных номеров и |
|  |  | ских характеристик | т.п. |

***7.3 Основные принципы стеганографического анализа***

**Определение 7.5**. ***Стеганоанализ* — процесс оценки перехвачен-ного контейнера на предмет наличия в нем скрытого (осажденно-го)сообщения**.

Понятно, что по сути *стегоаналитик* и *криптоаналитик* преследуют одинаковые цели.

**Основной *целью стеганоанализа* является моделирование стега-нографических систем и их исследование для получения качественных и количественных оценок надежности использования стеганопреобразо-вания, а также построение методов выявления скрываемой в контейнере информации, ее модификации или разрушения**.

Терминология *стеганоанализа* аналогична терминологии *криптоана-лиза*,однако имеют место некоторые существенные расхождения.Как мы ра-нее выявили (см., например, п.3), криптоанализ применяется с целью дешиф-рования криптограмм, а стеганоанализ — прежде всего, для выявления нали-чия скрытой информации.

Как мы выяснили, осаждение информации в электронном документе-контейнере требует изменений (перестройки) параметров или даже элемен-тов структуры последнего. Такие изменения могут выполнить роль "следа"', который сигнализирует о существовании встроенного сообщения, и, таким образом, основная идея стеганографии — скрытие факта существования сек-ретной информации — не будет выполненной.

Стеганоанализ на предмет наличия скрытой информации может при-обретать разные формы: обнаружение наличия (детектирование), извлечение и, наконец, удаление или разрушение скрытых данных. Кроме того, наруши-тель может поверх уже существующей скрытой информации встроить опре-деленную дезинформацию.

*По уровню обеспечения секретности стеганографические системы делятся на теоретически устойчивые, практически устойчивые и неустой-чивые.*

*Теоретически устойчивая стеганосистема* осуществляет скрытие111

информации только в тех фрагментах контейнера, изменение которых не превышает уровень, определяемый случайными факторами. При этом суще-ствует теоретическое доказательство того, что невозможно создать стеганоа-налитический метод выявления скрытой информации. Здесь мы можем обна-ружить некоторую схожесть со свойством криптосистемы на основе *однора-зовых блокнотов*.

*Практически устойчивая стеганосистема* выполняет такую модифи-кацию фрагментов документа-контейнера, которая может быть выявлена, но при этом известно, что на данный момент необходимые стеганоаналитиче-ские 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 сте-ганографического преобразования и усовершенствовать его таким образом, чтобы все изменения, внесенные в документ-контейнер, оказались бы в об-ласти теоретической или практической неразличимости.

**Определение 7.6. Стеганографическая система считается *взло-манной*,если стегоаналитику(или просто нарушителю)удалось,покрайней мере, доказать существование скрытого сообщения в перехва-ченном 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 со-общение, 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, как то:

* модификация скрытой информации.
* блокировка дальнейшие использования скрытой информация даже на

принципах санкционированного доступа.

Первые два этапа действия относят к *пассивным атакам на стегано-систему*,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 этомслучае злоумышленник имеет в своем распоряжении один или несколько за-полненных контейнеров (в последнем случае предполагается, что встраива-ние скрытой информации выполнялось тем же самым способом). Задача на-рушителя может заключаться в выявлении факта осаждения информации (наличия стеганоканала), а также в извлечении данных или определении ключа. Зная ключ, нарушитель имеет возможность анализа других стегано-сообщений.

112

*Атака на основании известного встроенного сообщения*.Этот типатаки характерен для систем защиты права интеллектуальной собственности, например, на текстовые документы или коды программ. Задачей анализа яв-ляется получение ключа. Если соответствующий скрытому сообщению за-полненный контейнер неизвестен, то задача является практически неразре-шимой.

*Атака на основании выбранного скрытого сообщения*.В этом случаенарушитель может предлагать для передачи свои сообщения и анализировать полученные при этом контейнеры-результаты.

*Адаптивная атака на основании выбранного сообщения*.Эта атакаявляется

частным случаем предыдущей. При этом нарушитель имеет возмож-ность выбирать сообщения для навязывания их передачи адаптивно, в зави-симости от результатов анализа предшествующих контейнеров-результатов.

*Атака на основании выбранного заполненного контейнера*.Этот типатаки более характерен для систем с использованием ЦВЗ. У аналитика есть некий анализатор заполненных контейнеров в виде "черного ящика" и не-сколько таких контейнеров. Анализируя выявленные скрытые сообщения, нарушитель пытается раскрыть ключ.

*Атака на основании известного пустого контейнера*.Если последнийизвестен нарушителю, то путем сравнения его с подозреваемым на присутст-вие скрытых данных контейнером он всегда может установить факт наличия стеганоканала. Как видим, данный тип атаки не имеет аналога в криптогра-фии.

*Атака на основании выбранного пустого контейнера*.В этом случаезлоумышленник 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 подозревае-мого со-общения от известной ему модели. Например, можно допустить, что биты в середине определенной части документа-контейнера имеют опреде-ленную взаимосвязь. Тогда отсутствие такой корреляции может служить сигналом о наличии скрытого сообщения.

*Вопросы для контроля и самоконтроля*

1. В чем состоит схожесть и в чем состоит основное отличие между криптографической и стеганографической системами?
2. Дать общую классификацию и соответствующую характеристику стеганографических систем.
3. В чем состоят особенности методов текстовой стеганографии? Дать сравнительную характеристику методов.

113

1. Разработать алгоритмы практической реализации выбранного (или по указанию преподавателя) метода текстовой или графической стега-нографии.
2. Охарактеризовать известные и/или предложить собственные ме-тоды стегоанализа.
3. Классифицировать атаки на стеганографические системы.

114

1. **Моделирование стеганографической системы**

**Определение 8.1**. **Абстрактно стеганографическая система обычно определяется как некоторое *множество отображений одного простран-ства* (множества возможных сообщений)в другое пространство(множе-ство возможных стегосообщений)**.Или наоборот.

Модель будем строить на основе следующих обозначений и положе-ний. Пусть M – это конечное множество сообщений, которые могут быть тайно размещены в контейнере: M **=** {*M1, M2,…,Mn*};

1. – это конечное множество всех допустимых контейнеров (файлов-контейнеров или документов-контейнеров): C = {*C1, C2,…* *Cp*}, причем *p>n;*

K – множество всех ключей, под которыми в общем случае будем по-нимать методы или алгоритмы осаждения сообщения в контейнере или иные операции по предварительному преобразованию осаждаемого сообщения или

выбору элементов контейнера для такого осаждения: K ={*K1, K2,…,Kz*}. Произвольное тайное сообщение *Mi*, можно скрыть в контейнере *Cj* при

использовании ключа *Km*: *Mi*  M*, i* = 1,2,…,*n*; *Cj*  C, *j* = 1,2,…,*p*; *Km*  K, *m* = 1,2,…,*z*.. Результатом такого типа преобразований будет заполненный кон-

тейнер (или стегосообщение) *Sq*, относящийся к множеству заполненных кон-тейнеров или стегосообщений S: S = {*S1, S2,…* *Sr*}, *q* = 1,2,…,*r*.

Дальнейшие рассуждения будут строиться на базе основных понятий, которые сформулированы в виде следующих определений.

**Определение 8.2**. **Функцию F, определенную на M *×* C *×* K со значе-ниями в S, будем отождествлять с осаждением или встраиванием сооб-щения *Mi* из множества M в контейнер *Cj* из множества C на основе клю-**

**ча из множества K*,* предусматривающего использование соответствую-щего алгоритма осаждения и пространственных (геометрических или иных) параметров элементов контейнера *Cj* множества C**:

|  |  |
| --- | --- |
| F***:*** M × C × K → S. | (8.1) |

Множество отображений

1. = {*F1, F2,…, Fl*},

графически можно представить в виде, показанном на рисунке 8.1.

Каждое конкретное отображение *Fw* ,где *w* = 1, 2,…, *l*,из множества

1. соответствует конкретному алгоритму или способу осаждения информа-ции *Mi* в контейнер *Cj* при помощи конкретного ключа *Kw.* Схематично представленное на рис. 8.1 отображение как раз поясняет такое взаимодейст-вие компонент системы.

Соотношение (8.1) формально описывает процедуру осаждения сооб-щения в контейнере на основе выбранного метода.

115

**Определение 8.3. Функцию F*- 1*, определенную на S × K\* (K \* ={*K*\**1,* *K*\**2,…,K*\**z*},в общем случае *Km ≠ K\*m* ; *Km*** **K, *K\*m*** **K\*; *m* = 1,2,…,*z*)со зна-чениями в M, будем отождествлять с извлечением тайного сообщения *Mi***

* **M из стегосообщения *Sq*** **S**:

|  |  |
| --- | --- |
| F-1: S × K\*→ M, C. | (8.2) |

Множеств F-1состоит из *l* элементов:

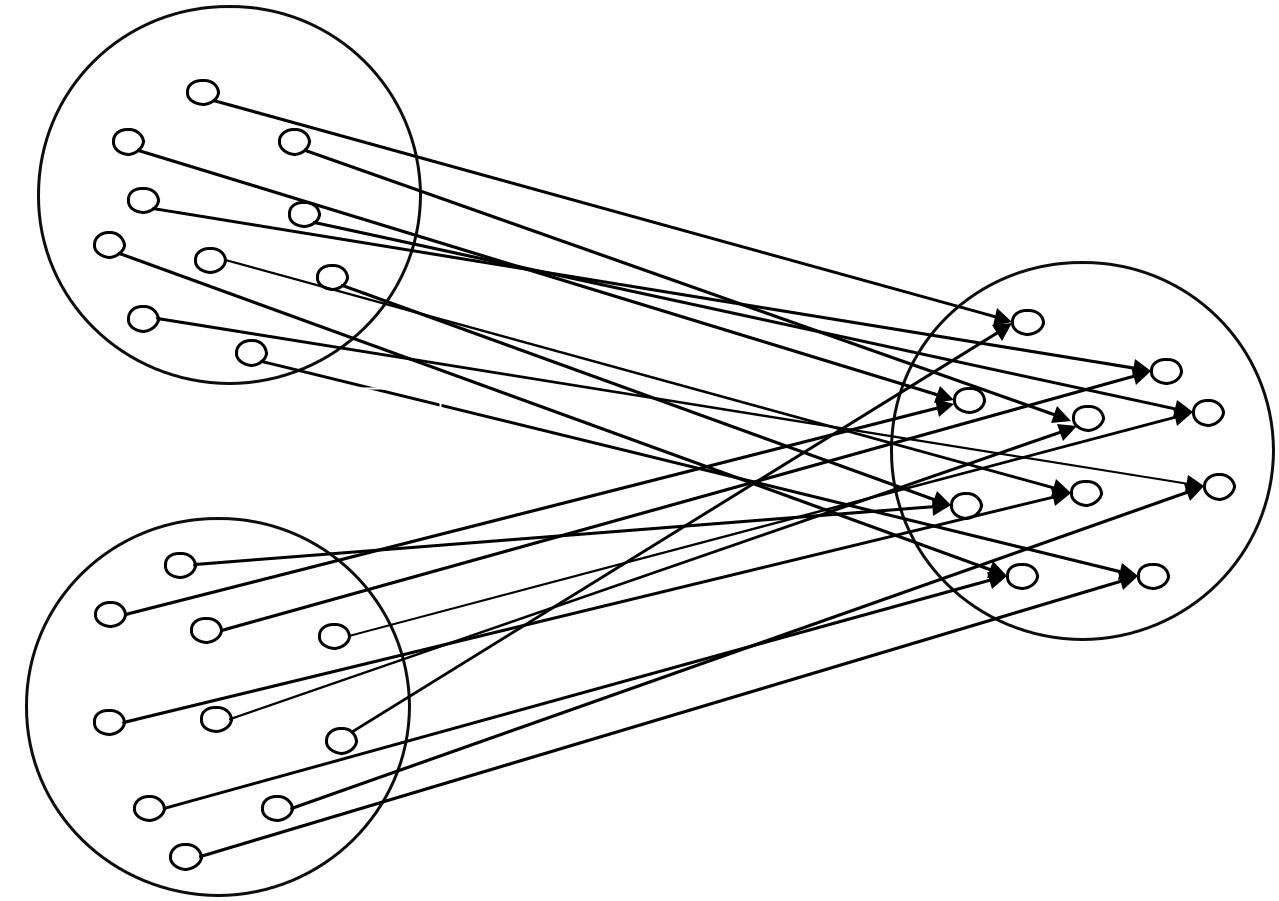
F***-1*** = {(*F-1*)1*,* (*F-1*)2*,…,* (*F-1*)*l* },

где каждому конкретному отображению *Fw* (*w* = 1, 2,…, *l*) соответствует фик-

сированный ключ *K\*w*  K\*.

Таким образом, выражение (8.2) определяет обратное по отноше-нию к (8.1) отображение, которое каждому элементу *Sq* множества S и фик-сированного элемента множества K\* ставит в соответствие элемент *Mi* мно-

жества M и элемент *Cj* множества C. Графическое представление такого ото-бражения иллюстрирует рис.8.2.



Множество открытых

текстов S

Множество

стеганограмм S

Множество

контейнеров C

Рисунок 8.1 – Графические представление отображения *Fw* на основе ключа *Kw* K

Соотношение (8.2) формально описывает процедуру извлечения со-общения из контейнера на основе того же выбранного метода. Соответствен-

116

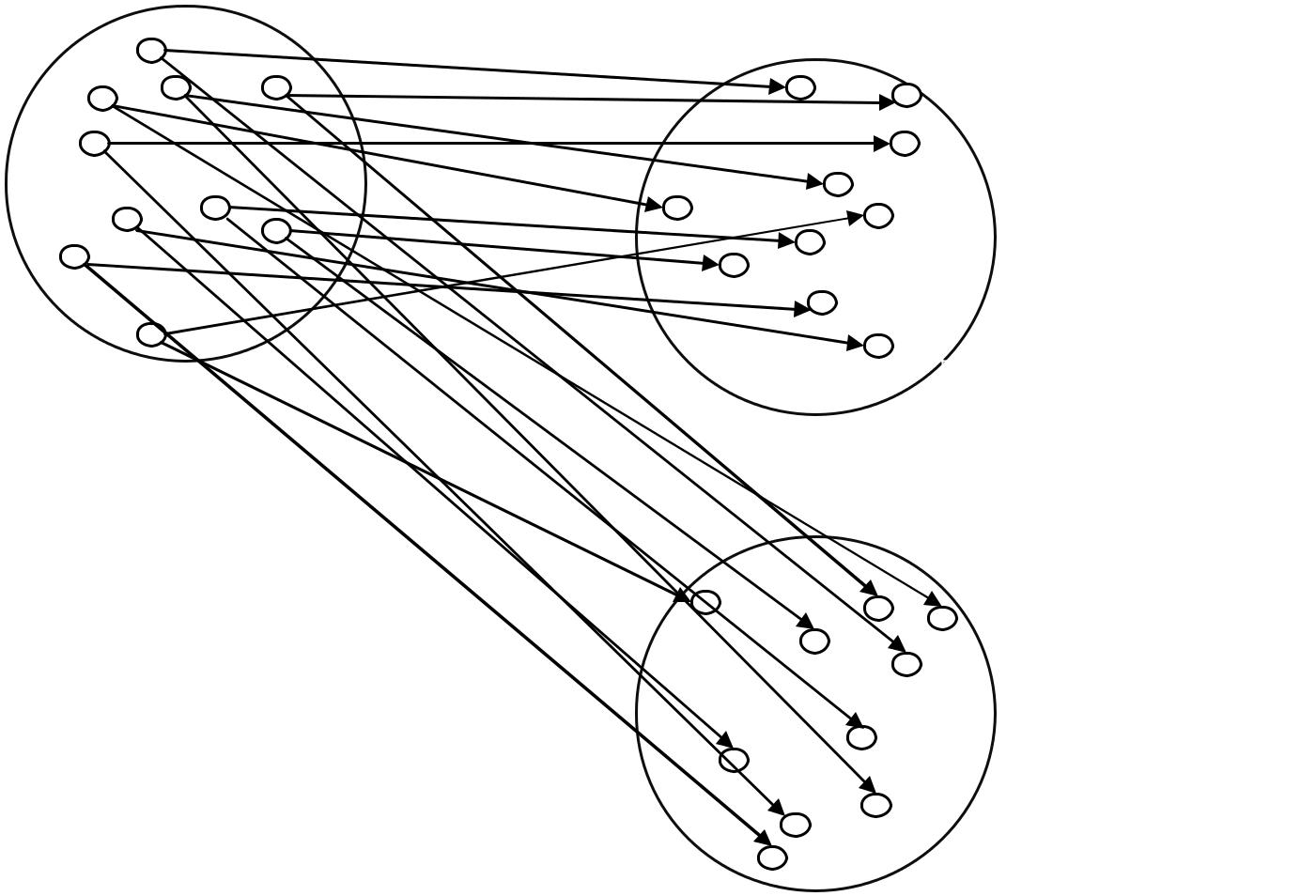
но каждое конкретное отображение (*F-1*)*w* , где *w* = 1, 2,…, *l*, из множества

F-1 соответствует конкретному алгоритму или способу осаждения информа-ции *Mi* в контейнер *Cj* при помощи конкретного ключа *K\*w.*

**Определение 8.4. *Коллизией стеганографического преобразования* (или пересечением) называем ситуацию, при которой различные эле-менты, относящиеся к одинаковым множествам: (*ma* *,c* *a,ka*) и (*mb* *,cb,kb*), формируют в результате одинаковые отображения, что формально мож-но записать в следующем виде**:

(*ma* *,ca,ka*) = (*mb* *,cb,kb*),

причем *ma≠* *mb,ca* *≠* *cb,ka* *≠* *kb,* либо неравенство выполняется хотя бы для од-ной из трех пар и 1 ≤ *а,b* ≤ *n*, *p*, *z*.



|  |  |  |
| --- | --- | --- |
|  | Множество открытых |  |
| Множество | текстов M |  |
|  |  |
| стеганограмм S |  |  |

Множество

контейнеров C

Рисунок 8.2 – Графические представление отображения *F-1w* на основе ключа *K\*w* K\*

**Определение 8.5. *Стеганографической системой* *∑* будем назы-вать совокупность сообщений M, контейнеров C, ключей K, стегосообще-ний (заполненных контейнеров) S и преобразований (прямого F и об-ратного *F*-1), которые их связывают**:

|  |  |
| --- | --- |
| ***∑*** = (M,C,K,S,F,F**-1**). | (8.3) |

117

Вернемся к обсуждению множества ключевой информации, K. Фор-мально этот тип множества, входящего в состав системы ***∑***, можно рассмат-ривать состоящим из некоторого числа непересекающихся подмножеств. Упомянутое число зависит от того, сколько факторов влияют на стойкость к взлому созданной системы. Под такими факторами (следуя постулату Керк-гоффса) будем понимать ключевую информацию. Иначе говоря, количество ключей.

Если тайна осаждения информации в контейнер обусловлена лишь ал-горитмом 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 алго-ритм или способ осаждения/извлечения информации24. И для таких стегано-графических систем справедливо формальное их описание в виде (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 зашифрова-

ния/расшифрования информации, относящейся к множеству M, при ее осаж-дении/извлечении;

б) алгоритмы или методы помехоустойчивого кодирования информа-

ции *Mi* (*Mi*  M) с целью обнаружения и/или исправления ошибок в этой ин-формации, появившихся в процессе передачи информации или ее хранения в этом контейнере;

и др. подобные.

Каждый из рассмотренных случаев (а-б) может быть отождествлен с использованием дополнительного (к базовому алгоритму осаждения/ извле-чения *Mi*) ключа.

Таким образом, мы можем говорить о *классе стеганосистем с допол-нительным ключом*.Каждый такой ключ *Km*дотносится к подмножеству до-

полнительных ключей K**д**.

**Определение 8.6. *Дополнительным ключом* Kд стеганографиче-ской системы будем считать конкретное секретное значение набора па-раметров криптографического или иного алгоритма, используемое для криптографического зашифрования/расшифрования сообщения, для помехоустойчивого кодирования/декодирования сообщения или иной дополнительной операции, используемой при осаждении/извлечении со-общения *Mi*; *K*д = {*K*1д, *K*2д,…, *Kt*д} в качестве дополнительного средства повышения стеганографической стойкости системы**.

Как отдельный тип ключа можно рассматривать псевдослучайные или иные метки, определяющие позиции или местоположение элементов контей-нера, которые подвергаются модификации при размещении в этом контейне-

1. 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*

118

ре определенного элемента (символа осаждаемого сообщения *Mi* (такие мет-ки, например, могут представлять собой позицию пиксела в матрице, форми-рующей изображение, текст или отдельный символ документа-контейнера при модификации его цветовой, яркостной или иной характеристики в про-цессе осаждения/извлечения тайного сообщения).

Значение такой информации в обеспечении общей стойкости стегано-графической системы к взлому сопоставимо со значением дополнительного

ключа, задаваемого из множества K**д**. Однако, исходя из физических принци-пов функционирования стеганосистем, ключевую информацию, относящую к принципам выбора местоположения для размещения конкретной единицы информации (бита, байта, символа и т.д.) сообщения *Mi*  M, *i* = 1,2,…,*n* в

документе-контейнере *Cj*  C, *j* = 1,2,…,*p* можно относить к множеству K. Поскольку генерация и использование дополнительных ключей прак-

тически не связаны с базовым алгоритмом осаждения/извлечения сообщения *Mi* в контейнере *Cj* с помощью ключа *Km*,целесообразно говорить о *двухклю-чевых стеганографических системах*.

**Определение 8.7. *Двухключевой стеганографической системой* *∑2* будем называть совокупность сообщений *M*, контейнеров C, ключей K, дополнительных ключей Kд, стегосообщений (заполненных контейнеров)**

1. **и преобразований (прямого F и обратного F-1), которые их связывают**:

|  |  |
| --- | --- |
| ***∑2*** = (M**,**C**,**K**,**K**д**,S**,**F**,**F**-**1). | (8.4) |

* соответствии с (8.4), стеганографические преобразования (осаждение и извлечение информации) для двухключевых систем в общем виде описыва-ются соотношениями:

|  |  |
| --- | --- |
| F***:*** M×С×K×K**д** ***→*** S, | (8.5) |
| F***-1:*** S***×***K***×***K**д** ***→*** M, C. | (8.6) |

Отображения, задаваемые выражениями (8.1) – (8.2) и (8.5) – (8.6), следует относить к числу *функциональных*, поскольку между каждым эле-ментом множеств, указанных в левой части (от стрелки) и в правой части должно существовать однозначное соответствие. В дополнение к этому от-метим, 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 канониче-

|  |  |
| --- | --- |
| ском виде: |  |
| F***-1:*** S***×***K ***→*** M , | (8.7) |
| F***-1:*** S***×***K***×***K**д** ***→*** M. | (8.8) |

Однако для систем, относящихся к задачам охраны прав интеллекту-альной 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 вос-

119

станавливать исходные элементы системы: тайную (авторскую) информацию

*Mi* Mи документ-контейнер *Cj* C,авторство которого должно быть под-тверждено с помощью *Mi*.

Теперь обратимся к множеству K**д**. В силу рассуждений, изложенных выше в пп. а) – б), это множество представим в виде непересекающихся под-множеств, число которых соответствует количеству типов используемых в системе дополнительных ключей (например, криптографических – им соот-ветствует подмножество K**дк**).

Если дополнительные ключи ограничиваются только криптографиче-скими ключами, то указанное множество K**д** и подмножество K**дк** совпадают.

Зафиксировав множество всех ключей K ={*K1, K2,…,Kz*}и множество

всех дополнительных ключей K**д** в виде криптографических ключей K**дк** ={*K1*дк*, K2*дк*,…Kt*дк} так, чтобы для всех *w* = 1, 2,…, *l* отображение *Fw* в соот-

ветствии с (8.5) однозначно задавалось этими ключами, т.е. *Fw*  F (и соот-

ветственно для (*F-1*)*w*  F***-1*** – для (8.6)), можем формально переписать это в следующем виде для одного фиксированного набора ключей: *Kw*дк и *Kw*:

|  |  |  |  |
| --- | --- | --- | --- |
| *Kw*дк | *Kw* |  |  |
| *Fw****:*** *Mi → Mi'* ; *Mi'* ***×*** *Cj* ***→*** *Sw* | | (8.9) |  |
| и | *Kw*дк |  |  |
| *Kw* |  |  |
| (*F-1*)*w****:*** *Sw* ***→*** *Mi* | *'*, *Cj*; *Mi'* ***→*** *Mi*, | (8.10) |  |

где *w* = 1,2, …, *l*.

* последнем случае каждое из преобразований – (8.9) или (8.10) – можно рассматривать как аддитивный процесс. Тогда графическое представ-ление соответствующих отображений, например (8.9), строится с учетом то-

го, что первая часть *Fw(1)* отображения есть не что иное, как отображение множества открытых сообщений в множество криптограмм или шифрограмм

на основе, например, ключа зашифрования/расшифрования *Kw*дк (см. рис.

8.3).

Вторая часть отображения *Fw* ( *Fw(2)*) на основе (8.9) графически будет выглядеть так же, как на рис. 8.1, с той лишь разницей, что вместо элемен-тов множества открытых текстов M следует использовать множество M***'***:

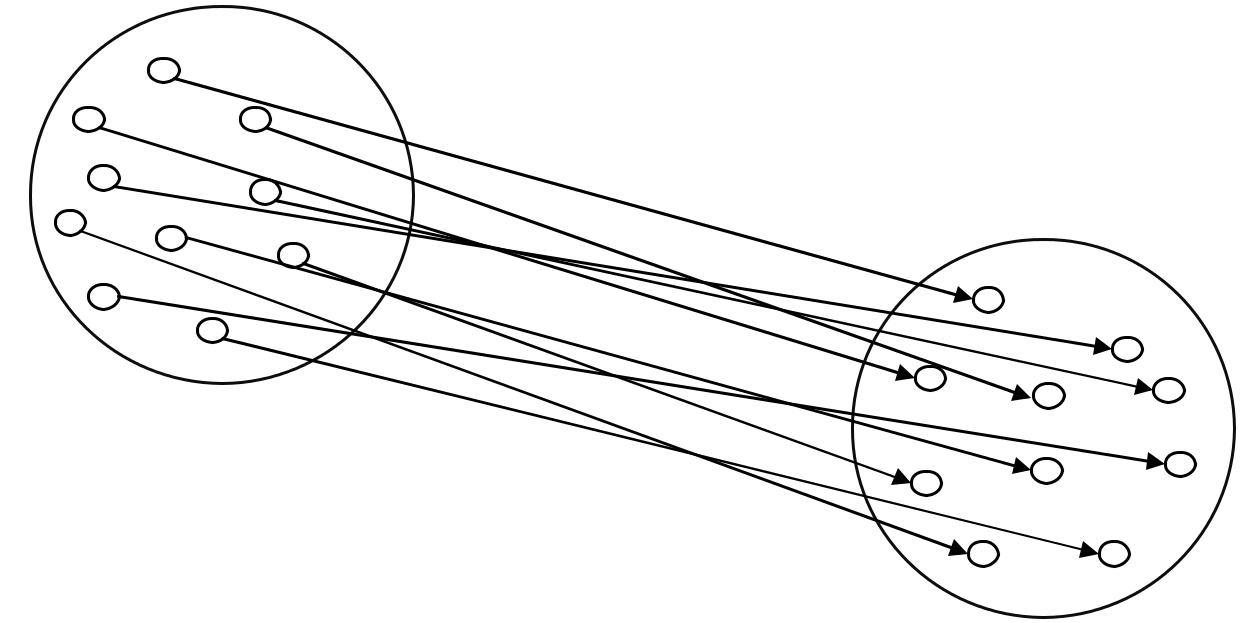
*Kw*

*Fw(2)****:*** *Mi'* ***×*** *Cj → Sw*.

Следует помнить о важном условии однозначности обратных преоб-разований: *мощности связанных между собой ключами преобразований* *множеств* MиS *должны быть одинаковыми*.Если прямое и обратное крип-тографическое преобразования осуществляются на основе асимметричной системы, то ключевая информация при соответствующем прямом и обратном стеганографическом преобразовании будет разной. Это мы подробно рас-смотрели в п. 5.6. Дополнительно мы это обстоятельство предусмотрели, приняв *Km* *≠* *K\*m* ; *Km*  K, *K\*m*  K\*; *m* = 1,2,…, *z,* где в общем случае *Km* и *K\*m*

120

фиксированные значения ключа прямого и обратного преобразований соот-ветственно.



Множество открытых

текстов M

Множество

шифрограмм M***'***

Рисунок 8.3 – Графические представление первой части отображения

*Kw*дк

в двухключевой системе на основе (8.9): *Fw(1)****:*** *Mi* *→* *Mi'*

Для подобных систем выражения (8.10) и (8.11), строго говоря, долж-ны быть изменены:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (*Kw*дк)з |  |  | *Kw* |  |  |
| *Fw****:*** *Mi → Mi* | | *'*; *Mi'****×*** *Cj* ***→*** *Sw* | | (8.11) |  |
| и |  |  | (*Kw*дк)р |  |  |
| *Kw* |  |  |  |  |
| (*F*-1)*w****:*** *Sw* ***→*** *Mi* | *'*, *Cj*; | | *Mi'* ***→*** *Mi*, | (8.12) |  |
| поскольку ключ зашифрования не равен ключу расшифрования: | | | | (*Kw*дк)з ≠ |  |

(*Kw*дк)р; *Kw*дк = ((*Kw*дк)з,(*Kw*дк)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 физиче-ских особенностей рассматриваемых процессов, дополнительный ключ *Kw*дк (равно как и *Kw*д) из множества ключей K**дк** (K**д**) при осаждении информации используется перед основным процессом – осаждения, а при обратной опера-ции – после извлечения сообщения из контейнера. В случае же использова-ния *составного дополнительного ключа* (например, шифрование и помехо-

121

устойчивое кодирование сообщения *Mi*), состоящего в общем случае из *v* подключей, относящихся к соответствующим элементам подмножества K**д**:

K**д** = { *K*1**д**, *K***2д**, …, *K***vд**}; *K*1**д** = { *K*11д, *K*12д, …, *K*1n д}, …, ***K***v**д** = { *K* v1д, *K*v2д, …, *K*vmд},последовательность использования такого подключа при извлечении

информации будет обратной (или зеркальной) по отношению к процессу осаждения.

Обратимся сейчас к ключу, образованному множеством K. Мы гово-рили о том, что в простейшем случае к ключевой информации можно отнести метод или способ осаждения. Для *стеганографических систем передачи ин-формации* 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 текстово-го документа-контейнера – метод *Line-shift coding, Word-shift coding* или иной). В задачах охраны права интеллектуальной собственности на конкрет-ный электронный документ имеет значение лишь алгоритм осаждения.

Таким образом, одно подмножество ключей (K***1***) множества ключей K образуют типы контейнеров (K***1с***) и алгоритмы (способы) осаждения инфор-

мации *Mi*  M (K***1а***): K***1*** = {K***1с*** , K***1а*** }; K***1с*** = {*K1cx*}, *x* = 1, 2, … , *с*; K***1а*** = {*K1ao*}, *o*

* 1, 2, ... , *h*. Другое подмножество ключей (K***2***) множества K образуют ука-занные выше метки, определяющие в общем случае выбор элементов кон-тейнера для осаждения соответствующих элементов сообщения. Понятно, что такие ключи также могут зависеть от типа используемого контейнера и

используемого алгоритма осаждения. Для упрощения примем, что K***2*** = {*K21*,

*K21*, …, *K2d*}.

Понятно, что ключи K***1с*** и K***1а*** могут нас интересовать в качестве само-стоятельных параметров только при анализе (или вероятностной оценке) ус-тойчивости системы к взлому, поскольку выбор типа контейнера носит не случайный, а детерминированный характер при решении задач защиты ав-торского права (в частности). Поэтому при формальном описании рассматри-ваемых процессов будем использовать только объединенный ключ, относя-щийся к множеству K.

С учетом изложенного выражения (8.9) и (8.10) в общем случае при-мут вид:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | (*K1w*д) | (*K2w*д) |  | (*Kyw*д) | | | (*Kw*) |  |
| *Fw****:*** *Mi' → Mi1'*; *Mi1' → Mi2* | | | *'*;...; *Mi(v-1)* | | *'→ Miv'*; *Miv'* ***×*** *Cj* ***→*** *Sw*; | | | (8.13) |
| и | *Kw* | (*Kvw*д)\* |  |  | (*K2w*д)\* | | (*K1w*д)\* |  |
| (*F-1*)*w****:*** *Sw* | ***→*** *Miv'*,*Cj*;*Miv'* ***→*** *Mi(v-1)* | | | *'*; ...; *Mi2* | | *' → Mi1* | *'*; *Mi1'* *→* *Mi'*, | (8.14) |

122

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| где *Kw* = {*K1* | | | | *w*, *K2* | | | *w*}; *K1* | | *w*  K***1***, *K2* | *w*  K***2***; K = { K***1***, K***2***}; K**д** = { *K*1 | | | | **д**, *K***2** | **д**, …, *K***vд**} и | |
| *K1w*дK1 | | | **д**, *K2w* | | д  K2 | | | **д**; ….; *Kvw*д  ***K***v**д**; K**д\*** = { *K*1 | | | **д\***, *K***2** | **д\***, …, | *K***vд\***}и(*K1w* | | | д)\*  |
| K1 | **д\***, (*K2w* | д)\*  K2 | | | | **д\***; ….; (*Kvw*д)\*  ***K***v**д\***; *Fw*  ***F***; (*F-1*)*w*  ***F-1***. | | | | | | |  |  |  |  |
|  | Перепишем (8.13) и (8.14) в общем виде, соответствующем (8.5) и | | | | | | | | | | | | | | | |
| (8.6), однако учитывающем последовательность операций: | | | | | | | | | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  | ***F:*** M***×*** K**д*×***C***×*** K ***→*** S, | |  |  |  |  | (8.15) | |
|  |  |  |  |  |  |  |  |  | ***F-1:*** S***×***K***×***K**д\* *→*** M,C. | | |  |  |  | (8.16) | |
|  | С учетом формальной разницы между K**д** и K**д\*** выражение (8.4) окон- | | | | | | | | | | | | | | | |
| чательно запишем так: | | | | | | | | | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | ***∑2*** = (M**,**C**,**K**,**K**д**,K**д\***,S**,**F**,**F**-**1). | | | |  |  | (8.17) | |

* соответствии с (8.15) выбранные ключи из одного множества допол-нительных ключей (K**д**) используются для предварительного одно- или мно-гократного (*v*-кратного) преобразования осаждаемого в контейнер *Cj*  C со-

общения *Mi*  M, а выбранные ключи из другого множества ключей (K) ис-пользуются непосредственно при реализации операции осаждения. Соответ-

ственно, ключи, относящиеся ко множеству K**д\***, используются при обратном *v*-кратном преобразовании на приемной стороне стеганографической систе-мы, описываемой (8.17).

Таким образом, выражения (8.13) и (8.15) позволяют выполнить опе-рацию синтеза передающей части стеганографической системы, которую по определению 8.6 мы назвали *двухключевой*. Эта часть система будет состоять

из следующих блоков: *источник потока сообщений* *Mi*  M, который форми-рует определенное сообщение *Мi*; *источник потока* (*множества*) *пустых*

*контейнеров* (*документов-контейнеров*) C, формирующий определенный

контейнер *Сj*  C для осаждения в нем сообщения *Мi*; *блок предварительного* *преобразования* (шифрование,кодирование и т.д.)сообщения *Мi*; *источник*

*дополнительных ключей*,формирующий *v* ключей из множестваK**д**:K**д**= {*K*1**д**, *K***2д**,…, *K***vд**}; *K*1**д**= { *K*11д, *K*12д, …, *K*1nд},…, *K*v**д**= { *K* v1д, *K*v2д, …, *K*vmд};ис-точник ключей K*,* *блок осаждения сообщения* *Mi* в контейнере *Cj* и формиро-вания стегосообщения *Sw* на основе ключа *Kw* K, *w* = 1, 2,…, *l*; *Sw*  S = {*S1,*

*S2,… Sr*}.

Точно так же на основе соотношений (8.14) и (8.16) можем синтезиро-вать приемную часть стеганографической системы (8.17). При этом необхо-димо принять во внимание обратный порядок (по отношению к вышеописан-ному) использования ключевой информации и формальную разницу в значе-ниях дополнительных ключей (K**д\***) в сравнении с K**д**.

* 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

123

шумов; изменение формата документа-контейнера: например, изменение шрифта текстового документа-контейнера и др.), которые приводят к изме-

нению стегосообщения *Sw*  S, в виде параметра *Sw*.

Структурная схема синтезированной двухключевой стеганографиче-ской системы, удовлетворяющей (8.4), (8.13), (8.15), приведена на рис. 8.4.

В этом случае, строго говоря, выражение (8.16) следует представить в

|  |  |
| --- | --- |
| виде |  |
| ***F-1:*** (S+*S*)***×***K***×***K**д\* *→*** M,C. | (8.18) |

Последнее будет справедливо для *робастных стеганографических* *систем*,т.е.для стегосистем,устойчивых к непреднамеренным воздействиямна элементы этих систем.

*Вопросы для контроля и самоконтроля*

1. Представить и пояснить структурную схему стеганографической системы: а) одноключевой, б) двухключевой.
2. Дать определения и формально описать стеганографические сис-темы: а) одноключевую, б) двухключевую.

124

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Источник | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | Источник | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | сообщений | | | |  |  |  |  |  |  | Интруз |  |  |  |  |  |  |  | Контейнер, Cj |  |  |
|  |  |  |  |  |  |  | контейнеров | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | *Mi* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *Cj* | | | |  |  |  |  | *Sw*+*ΔS* | *w* |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Блок | | | |  |  |  |  |  |  |  | ′ |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | *Mi*′ | | | |  |  | Блок осаждения | | | |  | *Sw* |  | Канал передачи |  |  | Блок извле- | |  | *Mi* | Блок обратного | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | предварительного | | | |  |  |  |  |  | (хранения) |  |  | чения | |  |  |  | преобразования |  |  |
|  | преобразования | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *Kw* |  | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Помехи |  |  |  |  |  |  |  |  |  |  |
|  |  |  | *Kw*д | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | Источник | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | ключей, *K* | | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Источник | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | дополнительных | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | ключей, *K*д | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

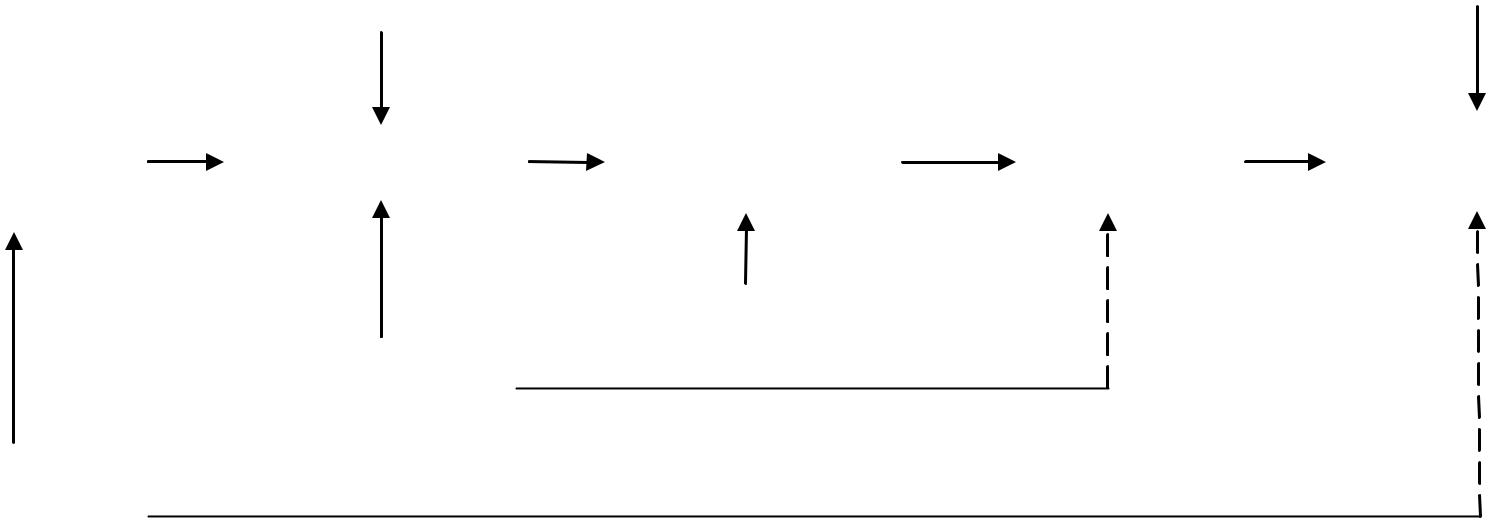
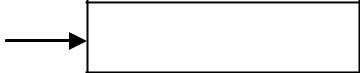


Рисунок 8.4 Структурная схема двухключевой стеганографической системы

Получатель



125

**ГЛАВА IV**

**ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ**

1. **ЭЦП на основе симметричной криптографии**

Электронная цифровая подпись (ЭЦП) является элементом крипто-графического преобразования информации. Однако учитывая специфиче-скую роль этого компонента информационных систем в общем комплексе за-дач, которые учитываются и решаются в предметной области, авторы посчи-тали целесообразным рассмотреть некоторые важные вопросы создания (ге-нерации) 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. По-добное действие может преследовать две цели. Во-первых, получатель кор-респонденции 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, обяза-тельств и т.д. С внедрением электронного документооборота (в том числе и конфиденциального) особо актуальной стала проблема установления под-линности и авторства безбумажной документации, что мы обсуждали в пре-дыдущей главе.

При всех преимуществах современных криптосистем они не позво-ляют обеспечить аутентификацию данных. Поэтому средства аутентифика-ции должны использоваться в комплексе с криптографическими алгоритма-ми. Электронная цифровая подпись (ЭЦП) как раз и является тем инструмен-том, который призван решать отмеченные задачи.

Понятие «электронная цифровая подпись» было введено в 1976 году У. Диффи и М. Хеллманом, высказавшими предположение о ее существова-нии. В 1977 году Р. Ривест, А. Шамир и Л. Адлеман разработали криптоси-стему RSA, которую выше мы достаточно подробно проанализировали и ко-торую можно было использовать для создания примитивных ЭЦП. Вскоре после RSA разработаны алгоритмы цифровой подписи И. Рабина и Р. Мерк-ле. В 1984 году Ш. Гольдвассер, С. Микали и Р.Ривест сформулировали тре-бования 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 получа-ется в результате криптографического преобразования электронных данных документа с использованием [личного ключа ЭЦП](http://edoc.by/eds#pk);

126

* *контролировать целостность* сообщения;ЭЦП вычислена на ос-

новании 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 бытьвыявлена путем операций сравнения соответствующих атрибутов подписан-ного и полученного адресатом сообщений;
* *доказать авторство* лица,подписавшего сообщение;создать кор-ректную ЭЦП можно, лишь зная закрытый ключ, известный только его вла-дельцу (лицу, подписавшему документ).

Важнейшими отличительными особенностям ЭЦП являются:

* ЭЦП представляет собой бинарную последовательность (в отличие от графического образа, каковым является подпись от руки);
* указанная бинарная последовательность зависит от содержания подписываемого сообщения.

Определим ЭЦП в следующем виде.

**Определение 9.1. *Электронная цифровая подпись* – последова-**

**тельность символов, являющаяся реквизитом электронного документа, зависящая от содержания этого документа и предназначенная для под-тверждения 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; упоминавшая выше в данном разделе ЭЦП на основе RSA относится именно к этому клас-су;

 на основе алгоритмов асимметричного шифрования и хеш-функции; это наиболее распространенная схема.

Общая структура подписанного электронного документа *М* представ-ляет собой, как правило, конкатенацию этого документа и ЭЦП. Кроме этих двух элементов, интегральный документ может содержать некоторую слу-жебную информацию (дата, время отправки или различные данные об отпра-вителе), как это схематично показано на рисунке 9.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Сообщение *М* | + | ЭЦП |  | Служебная |  |  |
|  | информация |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Рисунок 9.1. Структура документа с ЭЦП

Важное свойство цифровой подписи заключается в том, что еѐ может проверить каждый, кто имеет доступ к *открытому ключу* еѐ 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 ещѐ кому-то, кто тоже в

127

состоянии проверить эту подпись. Например, сторона *А* может переслать стороне *В* электронный чек. После того как сторона *В* проверит подпись сто-роны *А* на чеке, она может передать его в свой банк, служащие которого так-же имеют возможность проверить подпись и осуществить соответствующую денежную операцию.

И, наконец, еще одна особенность. Обычно подписываемые с помо-щью ЭЦП сообщения не шифруются. Они пересылается в исходном виде. Их содержимое не защищено от нарушения конфиденциальности. Путѐм совме-стного применения известных алгоритмов шифрования и ЭЦП можно созда-вать сообщения, которые будут и зашифрованы, и подписаны. Для этого ав-тор сначала должен добавить к сообщению свою сгенерированную ЭЦП, а затем — зашифровать получившуюся в результате пару (состоящую из само-го 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 доку-ментов, 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* обеспечи-вает собственно симметричный ключ, используемый двумя сторонами в про-цессе обмена сообщениями. Этот ключ согласован сторонами, известен (должен быть известен) только этим абонентам. Хотя в дополнение к обыч-ным образом зашифрованному сообщению не формируется ЭЦП, как само-стоятельный элемент послания (см. рис. 9.1) или просто шифртекста, как ви-дим, формально другая сторона получает вместе с сообщением все необхо-димые вышеперечисленные доказательства.

Первыми, кто обратил внимание на возможность симметричной схе-мы 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 *М*, т.е. по размеру ЭЦП может превосходить подпи-сываемый документ на несколько порядков. В силу этого серьезного недос-татка идея не нашла практического применения.

128

Однако, если предусмотреть наличие в системе третьего лица — ар-битра или посредника (*П*), – пользующегося доверием обеих сторон, то можно избежать указанного недостатка.

Основной алгоритм состоит в следующем.

1. Посредник *П* вырабатывает для *А* и *В* разные (сеансовые, например)

ключи: *КА* и *КВ*.

1. Абонент *А* шифрует свое сообщение *М* ключом *КА* и отсылает его посреднику: *С* = *ЕКА* (*М*).
2. *П* расшифровывает *С* ключом *КА* (*М* = *DКА* (*С*)),тем самым извлекаетсообщение *М*. Присоединяет к этому сообщению подтверждение того, что автором его является абонент А (обозначим эту часть нового сообщения *МД*). Таким образом сформирован конкатенированный документ *М’*= *М* || *МД*. По-средник шифрует *М’* ключом *КВ*: *С1* = *ЕКВ* (*М’*). Зашифрованное сообщение *С1* отсылается абоненту *В*.
3. Абонент *В* расшифровывает *С1* ключом *КВ* (*М’* = *DКВ* (*С1*)), тем са-мым извлекает сообщение *М’*. Из этого сообщения он получает оригинал сообщения *М* и подтверждение тому, что автором этого сообщения является абонент **А**.

Таким образом, здесь выполнены все функциональные требования,

присущие ЭЦП:

1. Подпись достоверна (*П* – гарант).
2. Подпись неподдельна, т.к. только *А* и *В* знают ключи, использовав-шиеся 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. Дать определение ЭЦП.
2. Основные функции ЭЦП.
3. Общие и отличительные признаки собственноручной и электронной

подписи.

1. Описать алгоритм ЭЦП с посредником. В чем состоят основные особенности алгоритма.

129

**10 ЭЦП на основе асимметричной криптографии**

***10.1. Особенности ЭЦП на основе алгоритмов асимметричной криптографии***

Асимметричные схемы ЭЦП относятся к криптосистемам с открытым ключом. Однако между алгоритмами асимметричной криптографии, которые мы проанализировали в п. 5.6, рассматриваемыми сейчас схемами формиро-вания подписей существует одно важное отличие. В отличие от асимметрич-ных алгоритмов шифрования, в которых зашифрование производится с по-мощью открытого ключа получателя, а расшифрование — с помощью закры-того ключа получателя, в схемах цифровой подписи подписывание произво-дится с применением закрытого ключа отправителя, а проверка — с приме-нением открытого ключа отправителя. Таким образом, следует запомнить:

***асимметричная криптография (передача зашифрованных сообщений) ос-нована на использовании ключей получателя, а асимметричная схема ЭЦП – ключей отправителя***.

При этом стойкость ЭЦП к подделыванию (криптостойкость) опреде-ляется теми же факторами, что и криптостойкость алгоритмов зашифрова-ния/расшифрования сообщений: чтобы применение ЭЦП имело смысл, необ-ходимо, чтобы вычисление легитимной подписи без знания закрытого ключа было вычислительно сложным процессом. Обеспечение этого во всех асим-метричных алгоритмах цифровой подписи опирается на известные нам вы-числительные задачи:

* *дискретного логарифмирования,*
* *факторизации*,то есть разложения числа на простые множители.Для ускорения процесса генерации подписи, т.е. для уменьшения объ-

ема вычислительных операций обычно принято подписывать не само сооб-щение *М*, а его образ, который получают путем вычисления хеша (*хеширова-ния*;иногда пишут *хэширования*)или *хеш-функции* от сообщения *М*: *h(M)*.Далее рассмотрим некоторые из известных алгоритмов хеширования.

***10.2 Основные понятия из области хешировании сообщений***

**Определение 10.1. *Хеширование* (англ. *hashing* ) – это преобразова-ние входного массива данных определенного типа и произвольной дли-ны в выходную битовую строку фиксированной длины. Такие преобра-зования также называются *хеш-функциями* или *функциями свертки*, а их результаты называют *хешем* или *дайджестом* сообщения (англ. *mes-sage digest* )**.

Существует и иное, более общее определение.

**Определение 10.2. *Хэш-функция* – математическая или иная функция, которая принимает на входе строку символов переменной**

130

**(произвольной) длины и преобразует ее в выходную строку фиксирован-ной (обычно – меньшей) длины, называемой *значением хеш-функции***.

Хеш-функцию можно использовать для преобразования произвольно-го входного текста в подходящий формат. Стоит также заметить, что исполь-зование хеш-функции не обязательно при вычислении ЭЦП, а сама функция не является частью алгоритма ЭП, поэтому хеш-функция может использо-ваться любая или не использоваться вообще.

Принято считать, что хорошей, с точки зрения практического приме-нения, является такая хеш-функция, которая удовлетворяет следующим ос-новным условиям:

* + функция должна быть простой с вычислительной точки зрения;
  + функция должна минимизировать число *коллизий* – то есть ситуа-ций, когда разным сообщениям соответствует одно значение хеш-функции.

При этом первое свойство хорошей хеш-функции зависит, в основ-ном, 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 из-вестному аргументу *х*) и сложность обратного вычисления (вычисления**

* **по известному *h(x)*)**.

Как мы ранее отмечали, однонаправленность – важнейшее свойство многих криптографических алгоритмов.

Обратимся сейчас к *коллизиям*. Эти, понятные с логической и с семан-тической точек зрения термин и явление являются следствием того, что мно-жество возможных сообщений всегда будет превышать множество возмож-ных хеш-функций, поскольку длина последних ограничена (чаще всего эта длина составляет от 128 до 512 бит). Это означает, что коллизии неизбежны, по крайней мере, с теоретической точки зрения. Принято коллизии разделять на два типа.

**Определение 10.4**. ***Коллизией* *1-го р*ода считаем ситуацию, при ко-торой для данного сообщения *М* и для иного произвольного сообщения *М’* (*М* ≠ *М’*)имеем *h(М)* = *h(М’)*,вычисленные с использование одной итой же хеш-функции (или алгоритма хеширования)**.

Приведем для лучшего понимания простую аналогию. Из закрытой емкости с разноцветными шарами мы выбрали шар определенного цвета, на-пример, красного (*М*). Далее наугад достаем другой шар (*М’*). 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 р*ода считаем ситуацию, при ко-торой для двух произвольных сообщений *М* и *М’* (*М* ≠ *М’*) имеем *h(М)* = *h(М’)*,вычисленные с использование одной и той же хеш-функции(илиалгоритма хеширования)**.

131

Следуя вышеописанной аналогии, в данном случае одновременно из-влекаются два шара. Коллизия наступает при их одинаковой окраске.

Рассматриваемая схема ЭЦП состоит из трех основных шагов:

* генерация ключа;
  + формирование подписи; для заданного электронного документа *М* (с помощью закрытого ключа) вычисляется подпись;

• проверка (верификация) подписи; для данных документа и подписи

* помощью открытого ключа определяется действительность подписи.

Для понимания сущности хеширования целесообразно проанализиро-вать реальный алгоритм. Например, MD4.

***10.3 Однонаправленные хеш-функции и алгоритмы хеширования***

***10.3.1 Алгоритм хеширования MD4***

Алгоритм был спроектирован Р. Ривестом (в то время – профессором Массачусетского Технологического Института) в 1990 г. Описание алгоритма

* пример реализации можно найти в документе ***RFC 1320***25. Это один из ал-горитмов целого семейства *MD* (message digest)**.** К этому семейству принад-лежат 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 произ-вольной длины. Предполагается, что сложность построения двух сообщений с одинаковым дайджестом имеет порядок 264 операций, и что сложность по-строения сообщения по заданному цифровому дайджесту имеет порядок 2128 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).

25 Rivest R., The MD4 Message Digest Algorithm, RFC 1320, MIT, 1992

132

*Стадия 1. Расширение входного сообщения*.Сообщение расширяетсятаким образом, чтобы его длина (в битах) была конгруэнтна 448 по модулю 512.То есть сообщение расширяется до размера так, что ему недостает всего 64 бита, чтобы иметь длину кратную 512. Расширение сообщения произво-дится 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 10-1*.Пусть дано сообщение«мир».Его длина(*l*)в битах рав-на 24 (3\*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: единствен-ный 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 бит.

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 би-тами? Они предназначены для дальнейшего расширения сообщения, теперь уже до размера, кратного размеру блока, т. е. 512бит. В этих 64 битах следует указать двоичное представление длины исходного сообщения («мир»). Еще раз отметим, что эта длина составляет 24 бита (3\*8). 64-разрядное представ-ление величины *l* (то есть длины сообщения до того, как оно было расшире-но) добавляется в конец результата, полученного на предыдущем шаге. При-чем, вначале дописывается младшее 32-разрядное слово, а затем старшее. В том случае, если величина *l* больше, чем 264, то добавляются только младшие

1. бита величины *l*.

Для рассматриваемого нами примера значащими (младшими) двоич-ными 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 сооб-щения,* которое имело длинно24бита.

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 зна-ем, что расширенная длина сообщения кратна 512. Это означает, что в таком формате дальнейшей обработке подвергается сообщение длиной *n*\*512 бит, где *n* – целое число, равное или большее 1.

На этой стадии каждый 512-битный блок разделяется на 16 32-разрядных слов (16\*32 = 512). Именно 32-разрядное слово является основной информационной и структурной единицей рассматриваемого алгоритма.

*Стадия 3. Инициализация констант*.Для вычисления цифрового дай-джеста используются 4 переменных (буфер размером в четыре слова), обо-

133

значаемых символами *A, B, C, D*. Таким образом, здесь каждое из *A, B, C, D* может отождествляться с *32-разрядным регистром*. Эти регистры инициали-зируются следующими *шестнадцатеричными* величинами или начальными константами (младший байт указан первым):

*A*: 01 23 45 67

1. 89 ab cd ef
2. fe dc ba 98
3. 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 дай-джест) обрабатываемого блока, а в конце обработки – итоговый хеш сообще-ния. Каждый регистр содержит 32-разрядную беззнаковую величину, поэто-му *размер хеша будет равен* *128* (32\*4=128) *битам*.

Каждый блок данных *Mj* преобразуется в 3-х раундах. Схематично это преобразование представлено на рисунках 10.1 и 10.2.

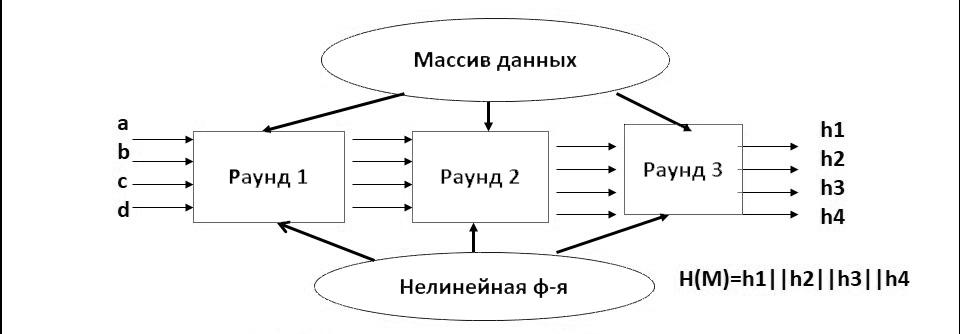


Рисунок 10.1 – Общая схема хеширования MD4

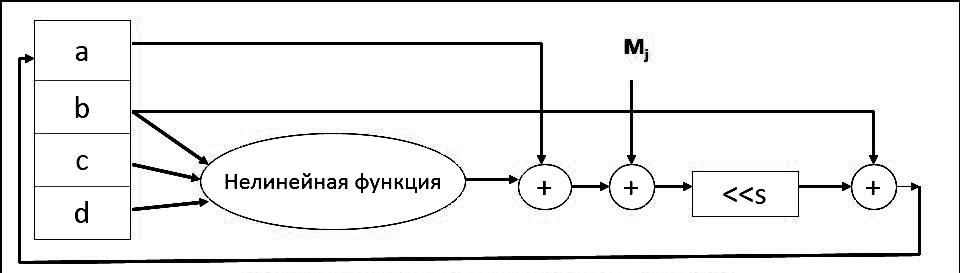


Рисунок 10.2 – Схема хеширования блока данных *Mj*

134

* 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)*:

*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),* | (10.1) |
| *H(x,y,z) = x XOR y XOR z.* |  |

Вспомним, что, например, *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 (табл. 10.1.)

Таблица 10.1. Таблицы истинности используемых функций

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | *y* | *z* | ***F*** |  | *x* | *y* | *z* | ***G*** |  | *x* | *y* | *z* | ***H*** |
| 0 | 0 | 0 | **0** |  | 0 | 0 | 0 | **0** |  | 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **1** |  | 0 | 0 | 1 | **0** |  | 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **0** |  | 0 | 1 | 0 | **0** |  | 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **1** |  | 0 | 1 | 1 | **1** |  | 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **0** |  | 1 | 0 | 0 | **0** |  | 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **0** |  | 1 | 0 | 1 | **1** |  | 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **1** |  | 1 | 1 | 0 | **1** |  | 1 | 1 | 0 | **0** |
| 1 | 1 | 1 | **1** |  | 1 | 1 | 1 | **1** |  | 1 | 1 | 1 | **1** |

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 преобра-зования. Эти операции используют определенные выше функции *F*, *G* и *H* и выражаются следующими формулами (с учетом (10.1)):

*a = (a + F(b,c,d) + X[k]) <<< s*

*a = (a + G(b,c,d) + X[k] + 5A827999) <<< s* (10.2)

*a = (a + H(b,c,d) + X[k] + 6ED9EBA1) <<< s*,

где *a,b,c,d* обозначают регистры *A, B, C, D*; *F, G* и *H* – рассмотренные выше функции (9.1); знак <<< - операцию циклического сдвига влево; *s* – количе-ство разрядов сдвига; *X[k]* – 32-разрядное слово, *k* = 0,1,2,…,15; во втором и третьем выражениях указаны 32-разрядные константы в 16-ричтной форме.

Последние выражения означают следующее. Например,

*a = (a + F(b,c,d) + X[k]) <<< s*

* применении к конкретным регистрам имеет следующий вид: *A = (A + F(B,C,D) + X[0])* <<< 3,

135

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 обрабаты-ваемого 512-битного блока – строит новое (текущее) значение регистра А.

* применении к регистру D операция преобразования может выгля-деть 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, сле-дующую последовательность элементарных преобразований для каждого из

1. 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 сооб-щения (цифровым дайджестом) будет результат вывода содержимого регист-ров *A, B, C, D* , точнее говоря, конкатенация 4-х 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 рисунке 10.3.

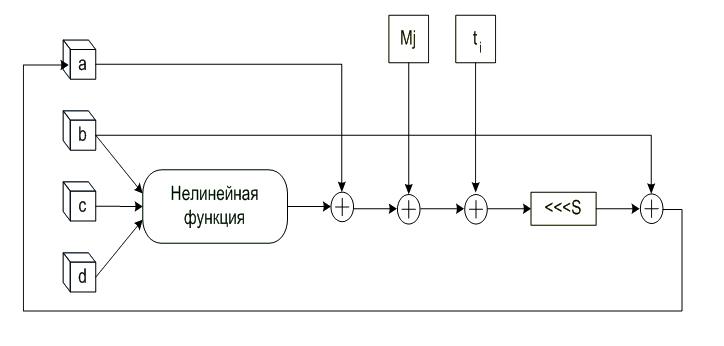


Рисунок 10.3. Общая схема алгоритма преобразования

блока данных на основе MD5

136

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 уникаль-ная константа для каждого раунда: *t*[*i*].

1. 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.

1. На каждом шаге используется значение, полученное на предыду-щем шаге. Это дает более быстрое изменение результата при изменении входных данных.
2. Для ускорения *лавинного эффекта* значения циклического сдвига в каждом раунде оптимизированы. Для этих же целей количество сдвигов раз-личается 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 хеширо-вания). 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 алгорит-мах MD, хеширование на основе SHA предусматривает выполнение предва-рительных преобразований.

* таблице 10.2 приведены основные характеристики алгоритмов клас-

са SHA.

Таблица 10.2. Характеристики алгоритмов хеширования SHA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Вариации | | |  | Размер |  | Промежуточный |  | Размер |  | Максимальная | |  | Размер |  | Количество |  | Найденные |  |
|  | выходного |  |  | блока, |  | длина входного | |  | слова, |  |  |  |
| алгоритма | | |  |  | размер хеша, бит |  |  |  |  | раундов |  | коллизии |  |
|  | хеша, бит |  |  | бит |  | сообщения, бит | |  | бит |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  | |  | |  |  |  |  | |  | |  |  |  |
| SHA-0 | | |  | 160 |  | 160 |  | 512 | 264 | − 1 |  | 32 |  | 80 |  | Есть |  |
|  |  |  |  | |  | |  | |  | |  |  | |  | |  |  |  |
| SHA-1 | | |  | 160 |  | 160 |  | 512 |  | 264 | − 1 |  | 32 |  | 80 | 252 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | операций |  |
|  |  |  |  | |  | |  | |  | |  |  | |  | |  |  |  |
| SHA- |  | *SHA-* |  | 256/224 |  | 256 |  | 512 |  | 264 | − 1 |  | 32 |  | 64 |  | Нет |  |
| 2 |  | *256/224* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 137 | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *SHA-* |  | 512/384 |  | 512 |  | 1024 |  | 2128 − 1 |  | 64 |  | 80 |  | Нет |
|  |  | *512/384* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Обратимся, к примеру, к алгоритму SHA-1. Как видим из таблицы, он выдает 160-битовое число. Для этого используются и инициируются изна-чально 5 32-разрядных констант:

*A*: 67 45 23 01,

1. fef c dab 89,
2. 98 ba dc fe, *D*:10 32 54 76, *E*:c3 d2 e1 f0.

Вначале эти константы копируются в соответствующие переменные:

*A* –в *а*, *В* –в *в* и т.д.

Главный цикл состоит из четырех раундов, каждый из которых вклю-чает по 20 операций (вспомним – в алгоритмах MD таких операций 16) . Ка-ждая такая операция предусматривает вычисление нелинейной функции над тремя переменными из набора *а,* *в,* *с,* *d, e*. После этого производятся опера-ции сдвига и сложения, аналогичные вышерассмотренным, с использованием константы *t*. В рассматриваемом алгоритме применяется следующий набор функций:

*F(x,y,x) = (x AND y) OR (NOT x AND z),*для *t* от0до19,

*G(x,y,z) = x XOR y XOR z,* для *t* от20до39, (10.3) *H(x,y,z) = (x AND y) OR (x AND z) OR (y AND z),* для *t* от40до59,

*I(x,y,z) = x XOR y XOR z,* для *t* от60до79.

Как видим, во всех рассматриваемых алгоритмах используются похо-жие функции.

Характеризуя криптостойкость рассмотренных алгоритмов, Б. Шнай-ер (в книге: Прикладная криптография, М.: Триумф, 2003) делает вывод:

***алгоритм SHA, по сути, совпадает с алгоритмом MD4, отличаясь нали-чием расширяющего преобразования, дополнительным циклом обработки***

* ***улучшенным лавинным эффектом. Алгоритм MD5 – это улучшенный***

***MD4*.**

*Вопросы для контроля и самоконтроля*

1. Дать определение хеш-функции.
2. Что такое «однонаправленность» хеш-функций и какова роль это-го свойства хеш-функций в криптографии?
3. Что такое «коллизия»?
4. Дать общую характеристику алгоритмам хеширования семейства

MD.

1. Из каких основных стадий состоит алгоритм хеширования сооб-

щения?

138

1. Рассчитать общую длину хешируемого сообщения после предва-рительной стадии на основе алгоритма MD, если объем исходного сообщения составлял: 0; 484; 512; 1000; 2000; 16000 бит. Какова в каждом случае будет длина хешированного сообщения?
2. Представить и охарактеризовать структурную схему одного раун-да алгоритмов хеширования на основе MD4; MD5; SHA-1.

139

**11 ЭЦП на основе хеша сообщения**

***11.1 ЭЦП при использовании RSA***

* п. 9.1 мы упоминали о возможности реализации основных функций ЭЦП с помощью простого шифрования сообщения на основе RSA. Подтвер-ждение подлинности и целостности полученного сообщения, которое зашиф-ровано ключом отправителя, основывается исключительности на принадлеж-ности ключа шифрования отправителю сообщения (тайного ключа отправи-теля: чисел *d* и *n*). Это означает, следующее: если получатель зашифрован-ного сообщения после его расшифрования публичным ключом отправителя получает логически стройный документ, то это является гарантом целостно-сти документа и указанием на авторство. Как видим, в данном случае ЭЦП, как самостоятельный и дополнительный элемент к сообщению, не генериру-ется (аналогия с использованием посредника).

Один из классических алгоритмов генерации ЭЦП на основе RSA от-личается от рассмотренного случая лишь тем, что отправитель шифрует не само сообщение *М* (см. п. 5.6.2), а его хеш, *h(М)*. Понятно, что предваритель-но должна быть выполнена процедура генерации ключа отправителем.

*Генерация ЭЦП* или зашифрование *h(М)*:

*s = (h(М)) d mod n,* (11.1)

здесь пара чисел (*d* и *n*) – тайный ключ отправителя.

После этого число *s,* являющееся ЭЦП, присоединяется к сообще-

нию *М* и пересылается получателю*.*

*Верификация подписи.* Получатель имеет в«своем распоряжении»:сообщение *М*, подпись к нему, *s*, известный алгоритм хеширования. Вери-фикация подписи предусматривает выполнение последовательных опера-ций:

|  |  |
| --- | --- |
| *восстановление h(М)*: |  |
| *(s) e mod n = h’(М)*; | (11.2) |

*вычисление хеша* для полученного сообщения: *h’’(М)* (для строгостианализа применяем новое обозначение);

*сравнение хешей*: *h’(М)* и *h’’(М)*;только при равенстве сравниваемыхвеличин 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 под-робно об этой процедуре речь пойдет ниже). Злоумышленник не сможет подменить сообщение (точнее, ему будет очень трудно это сделать), по-скольку для этого необходимо вместо сообщения *М* подставить другое со-общение *М*’, имеющее такое же значение хеша (хеш-функции), что и для *М*, что является, как мы это подчеркивали выше, вычислительно трудной зада-чей. По этой же причине злоумышленник не сможет применить перехвачен-

140

ную подпись *s* для подписи другого документа, поскольку для другого до-кумента будет получено иное значение хеша.

Таким образом, все необходимые свойства подписи описанным алго-ритмом обеспечиваются, что же касается криптостойкости метода ЭЦП, то она определяется криптостойкостью используемого асимметричного крип-тографического метода и функции однонаправленного шифрования. Необ-ходимо отметить также, что само сообщение *М* передается в открытом виде. Для того, чтобы обеспечить конфиденциальность передаваемой в нем ин-формации, 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 отправи-теля подлежит не хеш сообщения *М*, а само сообщение *М*:

|  |  |
| --- | --- |
| *s = М d mod n.* | (11.3) |

Получателю направляется пара: *М* и *s*; как и выше, *s* – это ЭЦП.

Процедура *восстановления сообщения* получателем:

|  |  |
| --- | --- |
| *(s) e mod n = М’*. | (11.4) |

Подчеркнем отдельно важную деталь: **в алгоритме зашифрова-ния/расшифрования сообщения на основе алгоритма RSA и в алгорит-ме генерации/верификации ЭЦП на основе того же алгоритма исполь-зуются различные ключи: в первом случае – ключи получателя (от-крытый – 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 верифика-ции)**.

Такой же механизм использования ключевой информации свойстве-нен и другим подобным алгоритмам.

***11.2 Схемы ЭЦП на основе проблемы дискретных логарифмов***

Практически все известные алгоритмы ЭЦП на основе хеша сообще-ния являются разновидностями общей схемы цифровой подписи, исполь-зующей известную нам *проблему дискретного логарифма*. К означенному семейство относятся, в том числе ЭЦП *DSA*, *Эль-Гамаля* и *Шнорра*, которые мы и рассмотрим ниже.

Формально все указанные разновидности подписи описываются сле-дующими рассуждениями.

141

Выберем большие простые числи *p* и *q,* при этом *q* должно быть ли-бо равным *р-1*, либо простому множителю *р-1*. Затем выбираем число *g* в диапазоне от 1 до *р,* для которого выполняется условие:

*gq = 1 mod p.*

Все эти числа открыты и могут использоваться группой лиц.

|  |  |
| --- | --- |
| Открытым ключом является число *y*: |  |
| *y = gx mod q*, | (11.5) |
| тайным – *х* (для сравнения рекомендуем обратиться к п. 5.6.3). |  |
| Для создания ЭЦП сообщения *М* выбирается случайное число *k* < *q*; | |
| причем числа *k* и *q* – взаимно простые. |  |
| Далее вычисляется число *r*: |  |
| *r= gk mod p*. | (11.6) |
| *Обобщенное уравнение подписи* имеет вид: |  |
| *a\*k = b + c\*x (mod q)*. | (11.7) |
| И *обобщенное уравнение проверки подписи*: |  |
| *ra = gb\*yc (mod p)*. | (11.8) |

***11.2.1 Алгоритм цифровой подписи DSA***

* августе 1991 года американским институтом стандартизации

((NIST) был утвержден стандарт *DSS* (Digital Signature Standard – стандарт цифровой подписи) для использования в алгоритме ЭЦП *DSA* (Digital Signa-ture Algorithm – алгоритм цифровой подписи).

* алгоритме используются следующие параметры:

*p* –простое число длиной от64до1024битов(число должно быть кратно64); *q* *–* 160-битовый простой множитель *р-1*.

Далее вычисляется число *g* :

*g = v(p-1)/g mod p,*

где *v* *–* любое число, меньшее *р-1*, для которого выполняется условие:

*v(p-1)/g mod p > 1.*

Числа *p*, *q*, *v* могут использоваться группой лиц.

*Открытый ключ у* вычисляется в соответствии с выражением:

|  |  |
| --- | --- |
| *y = gx mod p;* | (11.9) |

*x < q*; *х* – *закрытый ключ*.

Понятно, что открытый ключ размещается в общедоступной базе данных.

142

Вспомним, что в алгоритме подписывается не само сообщение *М*, а его хеш: *h(М)*. Для хеширования сообщений используется алгоритм SHA.

После генерации ключей их обладатель может подписывать свои со-общения.

*Генерация ЭЦП*.Выбирается из условия описанного выше число *k*.

Подпись состоит из пары чисел, *r* и *s*:

|  |  |
| --- | --- |
| *r = (gk mod p) mod q*, | (11.10) |
| *s = k-1 (h(M) + x\*r)) mog q*. | (11.11) |

Как видим, соотношения (11.10), (11.11) незначительно отличаются от (11.6) и (11.7).

Для проверки подписи получатель подписанного сообщения (М, *r*, *s*)

выполняет вычисления:

*w = s-1 mod q,*

*u1 = (h(M)\*w) mog q,*

*u2 = (r\*w) mod q,* (11.12)

*а = ((gu1 \* yu2) mod p) mod q.*

Подпись считается достоверной, если *а* *= r*.

На рисунке 11.1 приведена общая схема алгоритма.

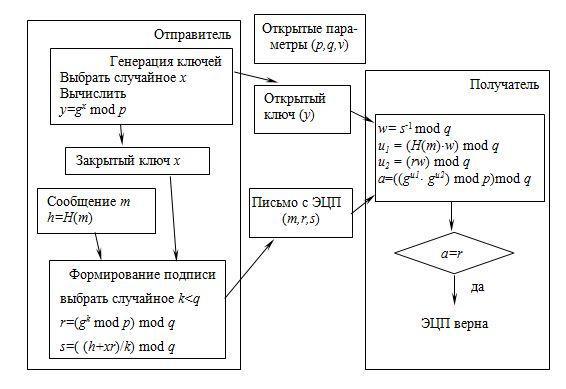


Рисунок 11.1 Общая схема алгоритма ЭЦП DSA

143

Быстродействие практической реализации рассматриваемого алго-ритма можно существенно увеличить с помощью предварительных вычис-лений. Обратим внимание на то, что *r* не зависит от подписываемого сооб-щения, *М*. Можно создать массив чисел *k* и рассчитать *r* для каждого *k*. Кроме того, можно вычислить значение *k-1* для каждого *k*. Далее брать оче-редную пару *r* и *k-1* для очередного сообщения *М*. По данным Б. Шнайера26 такие предварительные вычисления по времени сопоставимы с процедурой верификации подписи. При этом длительность операций вычисления *s* со-ставляет лишь несколько процентов от длительности всей процедуры гене-рации ЭЦП.

***11.2.2 Схема подписи Эль-Гамаля***

Схему Эль-Гамаля можно использовать как для шифрования (см. п. 5.4.3), так и для цифровых подписей. В сравнении, например, с ЭЦП на ос-нове RSA рассматриваемая схема обеспечивает более высокое быстродей-ствие.

Вспомним, что *ключ* состоит из четырех чисел. Для его генерации нужно выполнить следующие действия.

1. Выбирается простое число *р* и два случайных числа, меньших, чем *р*:числа *х* и *g*.
2. Далее вычисляется

*y = gx mod p*.

Открытый ключ: *y, g* и *р*; тайный ключ: *х*.

Чтобы подписать сообщение *М*, обладатель используемых для ЭЦП ключей должен выбрать, как и в предыдущей схеме, случайное число *k*, вза-имно простое с *р-1*. Затем вычисляется числа *а* и *b,* являющиеся цифровой подписью:

|  |  |
| --- | --- |
| *a = gk mod p*; | (11.13) |

для вычисления *b* с помощью расширенного алгоритма Евклида (см. листинг 4.1) решается уравнение

|  |  |
| --- | --- |
| *M = (x\*a + k\*b) mod (p-1)*. | (11.14) |

Для верификации подписи нужно убедиться, что выполняется равенство

|  |  |
| --- | --- |
| *ya \* ab = gM mod p*. | (11.15) |

1. Б. Шнайер. Прикладная криптография. – М.: Издательство ТРИУФ. – 2003. – 816 с.

144

Как видим, в *классическом алгоритме ЭЦП по схеме Эль-Гамаля ис-пользуется не хеш сообщения М, а подписываемое сообщение целиком*,кото-рое следует представлять также числом.

*Пример 11-1*.Положим,что *p*=11, *g*=2, *x*=8,вычисляем третье значение открытого ключа:

*y=gx mod p* = 28 *mod* 11 =3.

Таким образом, открытый ключ: *p*=11, *g*=2, *y*=3; закрытый ключ: *x*=8. *Подпись сообщения М* (*М*=5).Выбирается случайное числоk=9взаим-

но простое с *р-1*=10; вычисляется в соответствии с (11.13) первый элемент ЭЦП:

* *= gk mod p*= 29 *mod* 11= 6.

Второй элемент ЭЦП находим основе уравнения (11.14): *M = (x\*a+k\*b) mod (p-1)* = 5=(8\*6+9\*b) *mod* 10.

Решением будет число *b* =3.

Подписанное сообщение: *М*=5, *а*=6, *b* =3. *Проверка подписи* в соответствии с(11.15):(36 \* 63) *mod* 11 = 25 *mod* 11= 10.

* ЭЦП DSS и Эль-Гамаля для каждой новой подписи необходимо ис-пользовать новое значение *k*. Если третья сторона добудет два сообщения, подписанные с использованием одного и того же *k*, ее шансы на взлом закры-того ключа отправителя значительно возрастают. Это обстоятельство мы под-черкивали при рассмотрении схем шифрования по Эль-Гамалю.

***11.2.3 ЭЦП на основе схемы Шнорра***

На основе схемы К. Шнорра (Claus Schnorr) строятся протокол аутен-тификация и ЭЦП. При этом подписывается, как и в алгоритме *DSA*, не сооб-щение *М*, а его хеш-функция на основе *M*.

* алгоритме используются следующие открытые параметры: *p* –простое число в диапазоне от512до1024бит;

*q* –160-битовое простое число,делитель *p-1*;

любое число *z* (*z*≠1) такое, что

*zq=1 mod p*.

Числа *p, z, q* являются открытыми и могут применяться группой поль-зователей.

Выбирается число *х* *< q* (*х* является тайным ключом) и вычисляется от-

|  |  |
| --- | --- |
| крытый ключ: |  |
| *v=z-х mod p*. | (11.16) |

Секретный ключ имеет длину не менее 160 бит.

Для *подписи сообщения М* выбирается случайное число *k (k<q)* и вы-числяет число *а*:

|  |  |
| --- | --- |
| *а= zk mod p*. | (11.17) |

145

Далее вычисляется хеш от канкатенации сообщения *М* и числа *а*:

* *= h(M||x)*

и вычисляется значение

|  |  |
| --- | --- |
| *y = (k + s\*Н) mod q*. | (11.18) |

Получателю отправляются *М,* *Н, y*; понятно, что ЭЦП составляют два последних элемента этого сообщения.

*Проверка подписи:* получатель вычисляет

|  |  |
| --- | --- |
| *x’ = zy \* vН mod p*; | (11.19) |

затем он проверяет, рано ли *Н* хеш-значение от конкатенации *М* и *x’:*

*Н’ = h’(M||x’)*

Подпись достоверна, если *Н* *=* *Н’.*

*Существующий стандарт ЭЦП в Республике Беларусь* (СТБ34.101.45-2013) основан на схеме Шнорра (а также на эллиптических кривых)27.

*Вопросы для контроля и самоконтроля*

1. В чем состоит основное отличие криптосистемы переда-чи/получения зашифрованного сообщения передачи/получения подписанного сообщения?
2. На чем основана криптостойкость ЭЦП Эль-Гамаля, Шнорра и

DSA?

1. Представить схемы ЭЦП Эль-Гамаля, Шнорра, взяв за основу схему на рис.11.1.
2. Взяв ключевую информацию из примера 5-1, подпишите сооб-щение, состоящее из собственного имени алгоритмом RSA.
3. Взяв ключевую информацию из таблицы задания после п. 5.6.3, подпишите сообщение, состоящее из собственного имени алгоритмом Эль-Гамаля.
4. Российские стандарты ЭЦП (ГОСТ Р 34.10–2001 и актуальный - ГОСТ Р 34.10-2012) также основаны на эллиптических кривых

146

**12 ЭЦП на основе эллиптических кривых**

Мы достаточно подробно (см. п. 5.4.4) проанализировали особенности практического применения в криптографии эллиптических кривых, задавае-мых над полями Галуа. Для ясности вспомним важнейшие особенности.

Пусть задано простое число *p* > 3. Тогда эллиптической кривой *E*, оп-ределенной над простым конечным полем *Fp*, называется множество пар чи-сел (*x, y*) (точка с координатами (*x*, *y*) принадлежит *Fp*), которые удовлетво-ряют тождеству:

|  |  |
| --- | --- |
| *y2 = x3 + ax + b mod p,* | (12.1) |

где *a*, *b* ∈ *Fp* и (*4a3* + *27b2*) ≠ 0 mod *p*.

Кроме того, к эллиптической кривой добавляется бесконечно удален-ная точка *О*. Таким образом, точки, удовлетворяющие уравнению кривой *E*, и точка *О* образуют конечную абелеву группу.

Для точек эллиптической кривой определена операция сложения. Для двух точек, принадлежащих кривой *E*, *P*(*xp*, *yp*) и *Q*(*xq*, *yq*), точка *R*(*xr, yr*), яв-ляющаяся их суммой, также будет лежать на эллиптической кривой (см. рис. 5.4).

Графически удвоение точки можно получить, построив касательную к точке и отразив точку пересечения касательной с эллиптической кривой от-носительно оси OX. Отсюда очевидно, что можно определить операцию ум-ножения некоторой точки эллиптической кривой на целое число, которая по-зволяет определить точку *Q* = *k\*P* (точка *P*, умноженная на целое число *k*, обращается в точку *Q*).

Скалярное умножение осуществляется посредством нескольких ком-бинаций сложения и удвоения точек эллиптической кривой.

*Пример 12-1*.Некоторая точка25\**P* может быть представлена как25\**P* = 2\*(2\*(2\*(2\**P*)) + 2\*(2\*(2\**P*))) + *P*.

Именно с **операцией умножения точки на целое число напрямую** **связана идея, надѐжность и криптостойкость эллиптической криптогра-фии**.

Мы уже особо отмечали, что *задача вычисления дискретного лога-рифма на эллиптической кривой заключается в отыскании целого числа х по известным точкам P и Q = х\*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 *х*, а сообщаемым открытым ключом *–* *произведение х на базисную* *(генерирующую)* *точку* *G***.

Выбор базисной точки обусловлен тем соображением, чтобы ее поря-док (для примера см. таблицу 5.10) был достаточно большим 2254 < *n* < 2256. Вспомним: **точка** ***P*** **∈** ***E*** **называется точкой порядка** ***n*,** **если** ***n\*P*** **=** ***O***.

147

Для того, чтобы лучше понять практику использования рассматривае-мой технологии, проанализируем весь алгоритм, взяв за основу существую-щий стандарт ЭЦП в России.28

*Пример 12-2*.Будем использовать следующие обозначения:

*H(M)l* –множество всех двоичных векторов длиной *l* бит–множествохеш-функций подписываемых сообщений; *l* = 256 бит (либо 512 бит);

*V\** –множество всех двоичных векторов произвольной конечной длины; *p* –простое число, *p* > 3;

*M* –сообщение пользователя, *M* ∈ *V\**;

*a, b –* коэффициенты эллиптической кривой*;*

*P –* точка эллиптической кривой порядка *q*;в рассматриваемом стандарте *q* выбирается из условия*:* 2254 *< q <* 2256; *q*\**P* =*O* (точка *Р* соответствует гене-рирующей точке кривой и является открытым параметром))*;* *d –* целое число–тайный ключ подписи;0 < *d < q;*

*Q –* точка эллиптической кривой–ключ проверки подписи:

|  |  |
| --- | --- |
| *Q* = *d\*P*; | (12.2) |

* – цифровая подпись под сообщением *М*.

Используемая эллиптическая кривая задается соотношением (12.1). Координаты точек при *х1≠* *х2* вычисляются по формулам (5.17), а при выборе кривой следует руководствоваться рекомендациями, которые изложены в п. 5.4.4.2 настоящего пособия.

Как видим, каждый пользователь должен иметь тайный криптографи-ческий ключ (число *d*) и ключ проверки или верификации подписи: точкой эллиптической кривой *Q* с координатами (*xq* и *yq*), удовлетворяющей (12.2).

*Формирование (генерации) ЭЦП*.Состоит из6оперраций.

1. Вычислить хеш сообщения *М*: *h(M); h(M)* *∈* *H(M)l*.
2. Вычислить целое число *t*, двоичным представлением которого является *h(M)* и далее вычислить

|  |  |
| --- | --- |
| *e = t mod q*. | (12.3) |

Если *е* =0, то принять *е* = 1.

1. Выбрать случайное (псевдослучайное) число *k*; 0< *k* < *q*.
2. Вычислить точку (координаты точки) *С* кривой *Е*:

|  |  |  |
| --- | --- | --- |
|  | *С = k\*P,* | (12.4) |
| определить число *r*: | |  |
|  | *r = xc mod q,* | (12.5) |
| где *xc* *–* *х*-координата точки *С*. Если *r* = 0, *вернуться к шагу* *3.* | |  |
| 5. | Вычислить значение *s*: |  |
|  | *s = (r\*d + k\*e) mod q*, | (12.6) |
| при *s* = 0 | следует повторить шаг 3. |  |

28 ГОСТ 34.10-2012

148

1. Определить двоичное представление чисел r и s, конкатенация которых (*r||s*) и будет является цифровой подписью *ζ* под сообщением *М.*

*Верификация подписи*.

1. Выполняется операция, обратная по отношению к шагу 6 генерации ЭЦП: восстановление чисел r и s.
2. Вычисляется хеш полученного сообщения *М*: *h’(M*).
3. Повторяется шаг 2 режима генерации ЭЦП: вычисляется целое чис-ло *t*, двоичным представлением которого является *h’(M)* (формально это зна-чение может приводить к различным результатам здесь и при использовании (12.3)) и далее вычислить

|  |  |
| --- | --- |
| *e’ = t mod q*. | (12.7) |

Если *е’* =0, то принять *е’* = 1.

1. Вычислить

|  |  |
| --- | --- |
| *v = (e’)-1 mod q*. | (12.8) |

1. Определить значения двух чисел:

|  |  |  |
| --- | --- | --- |
| *z1 = s\*v mod n*, | *z2 = -r\*v (mod q).* | (12.9) |
| Здесь полезно обратиться к (5.14). |  |  |
| 6. Вычислить точку *С* эллиптической кривой: | |  |
| *С =z1\*P + z2\*Q* | | (12.10) |
| и определить значение числа *r’*: |  |  |
| *r’ = xc mod q.* | | (12.11) |
| 7. Подпись верна, если ***r’ =*** ***r***. |  |  |

Таким образом:

* **исходными данными алгоритма генерации ЭЦП являются: сооб-щение *М*, алгоритм и параметры вычисления хеш-функции сообщения *М*,значение тайного ключа отправителя, *d*,тип эллиптической кривой,случайное число *k*, с помощью которого определяется точка *С* кривой, точки эллиптической кривой: точка *P* порядка *q* и точка *Q*, удовлетво-ряющая (12.2)**;
* **исходными данными алгоритма верификации ЭЦП являются: по-лученное сообщение *М*, электронная подпись *ζ* сообщения, точка *P* по-рядка *n* и точка *Q*.**
  + анализируемом стандарте ЭЦП рассмотрен пример с конкретными числовыми значениями, который проанализируем здесь с некоторыми со-кращениями.

Параметру *р* присвоено следующее значение: *р* *>* 0.5\*1077 или точно

*p*=57896044618658097711785492504343953926634992332820282019728792003956564821041;

коэффициенты эллиптической кривой:

149

* = 7 и *b>* 0.4\*1077 или точно:

1. = 43308876546767276905765904595650931995942111794451039583252968842033849580414;

порядок группы точек эллиптической кривой: число, большее 0.5\*1077 или точно:

57896044618658097711785492504343953927082934583725450622380973592137631069619;

координаты точки *Р*: *xp*=2, *yp* *>* 4\*1076 или точно:

*yp* =4018974056539037503335449422937059775635739389905545080690979365213431566280; *q* =57896044618658097711785492504343953927082934583725450622380973592137631069619;

тайный ключ отправителя сообщения: *d* > 0.5\*1077 или точно:

*d* =55441196065363246126355624130324183196576709222340016572108097750006097525544;

ключ *Q* проверки подписи – координаты точки имеют следующие значения:

*xq* > 0.5\*1077или точно:

*xq* =57520216126176808443631405023338071176630104906313632182896741342206604859403;

*yq* > 1.7\*1077или точно:

*yq* =17614944419213781543809391949654080031942662045363639260709847859438286763994.

*Генерация ЭЦП*.

Пусть после выполнения шагов 1– 2 в соответствии с (12.3) было по-лучено следующее числовое значения *е*:

* = 20798893674476452017134061561508270130637142515379653289952617252661468872421;

на шаге 3 выбрано число *k*:

*k* =53854137677348463731403841147996619241504003434302020712960838528893196233395;

при этом точка *С* *=* *k\*P* имеет координаты:

*xc* =29700980915817952874371204983938256990422752107994319651632687982059210933395; *yc* =32842535278684663477094665322517084506804721032454543268132854556539274060910;

параметр *r* = *xc* *mod q* (12.5) принимает значение:

*r* =29700980915817952874371204983938256990422752107994319651632687982059210933395;

параметр *s* = *(r\*d+k\*e)(mod q)* соответственно равен:

1. = 574973400270084654178925310019147038455227042649098563933718999175515839552.

Цифровой подписью (*ζ*) будет бинарная последовательность, состоя-щая из конкатенированных величин (*r||s*), представленных в соответствую-щей системе счисления.

*Верификация подписи*.

На входе получатель подписанного сообщения имеет *М* и *ζ*, как ос-новные компоненты для верификации.

После выполнения третьего шага получим *е’* (12.7):

*е’ =* 20798893674476452017134061561508270130637142515379653289952617252661468872421;

как видим в процессе передачи подписанного документа его целостность не нарушена (*е’=е*);

параметр *v* = *(е’)-1* *(mod q)* принимает значение (12.8):

*v* =17686683605934468677301713824900268562746883080675496715288036572431145718978;

на очередном шаге алгоритма вычисляем:

150

*z1* =37699167500901938556841057293512656108841345190491942619304532412743720999759;

и

*z2* =1417199842734347211251591796950076576924665583897286211449993265333367109221;

точка *С* =*z1\*P+z2\*Q* (12.10) будет иметь координаты:

*xc* =29700980915817952874371204983938256990422752107994319651632687982059210933395; *yc* =32842535278684663477094665322517084506804721032454543268132854556539274060910;

наконец, вычисляем *r’=* *xc* *(mod q)* :

*r’* =29700980915817952874371204983938256990422752107994319651632687982059210933395.

Поскольку *r’* = *r*, подпись верифицирована.

Главным достоинством ЭЦП на основе ЭК является то, что они обес-печивают криптостойкость на существенно меньших по длине ключах (в сравнении со схемами, рассмотренными в р.11), что положительно отражает-ся на времени генерации и верификации ЭЦП. Например, ЭЦП на основе эл-липтических кривых с длиной ключа 160 бит 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?

1. Какую ключевую информацию используют отправитель и получа-тель подписанного сообщения на основе ЭК?
2. Используя тайный ключ (*d*) из таблицы задания после п. 5.6.3, под-пишите сообщение, состоящее из собственного имени алгоритмом на основе ЭК, параметры которой определяются в примере 5-10.
3. В чем по Вашему состоят преимущества схемы ЭЦМ на ЭК в срав-нении с традиционными? В чем состоят их недостатки?

151

**13. Атаки на ЭЦП и противодействие атакам**

***13.1. Характеристика основных типов атак***

Целью атак на цифровые подписи является, в конечном итоге, воз-можность их подделки или фальсификации. Известны модели атак на ЭЦП, описанные достаточно давно Ш. Голвассером, С. Мисали и Р. Ривестом,29 однако актуальные и в настоящее время. Здесь мы найдем много схожего с атаками на шифры (см. п. 3.5). Кратко охарактеризуем эти атаки.

*Атака с использованием открытого ключа*.Имеется в виду ключсубъекта, подписывающего документ. *Криптоаналитик* обладает только от-крытым ключом.

*Атака на основе известных сообщений*.В распоряжении криптоана-литика имеются некоторые ЭЦП и соответствующие им документы.

*Адаптивная атака на основе выбранных сообщений*.Криптоаналитикможет получить подписи электронных документов, которые он выбирает сам.

Основные результаты или цели описанных атак можно классифици-ровать следующим образом.

*Полный взлом ЭЦП*.Получение закрытого ключа отправителя подпи-санного 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 под-писи для документов, выбранных криптоаналитиком.

*Экзистенциальная подделка подписи*.Дает возможность получениядопустимой подписи для некоторого документа, не выбираемого криптоана-литиком.

Самой опасной атакой является адаптивная атака на основе выбран-ных 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. Бо-лее вероятен поиск *коллизий*, сущность которых мы рассмотрели при анализе хеш-функций. С учетом применения хеш-функций во многих основных схе-мах ЭЦП, нахождение коллизий для алгоритма подписи эквивалентно нахо-ждению коллизий для самих хеш-функций. Таким образом, в контексте об-суждаемой проблемы во возвращаемся к коллизиям первого и второго рода.

1. Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks// SIAM Journal on Computing. – 1988, 17(2). – P. 281—308.

152

*Коллизия первого рода эквивалентна экзистенциальной подделке, а коллизия второго рода — выборочной*.

*Подделка документа* (коллизия первого рода).Криптоаналитик под-бирает документ *М’* к данной подписи. Однако в подавляющем большинстве случаев такой документ может быть только один. Это оригинальный доку-мент *М*, для которого и создавалась данная ЭЦП. Причина здесь кроется в следующем. Во-первых, документ представляет из себя осмысленный, логи-чески стройный текст. Во-вторых. Текст документа оформлен по установ-ленной форме (как правило, каждый тип электронного документа в каждой стране должен создаваться на основе установленных правил, формируемых на основе национального законодательства; например, у нас в стране базо-вым документом является *ЗАКОН РЕСПУБЛИКИ БЕЛАРУСЬ «Об элек-тронном документе и электронной цифровой подписи» от 28 декабря 2009 г.*

* *113-З*)*.* Это значит,если для фальшивого набора байт(*M’*)и произойдетколлизия с хешем исходного документа (*М*), то должны выполниться 3 сле-дующих условия:
  + случайный документ должен соответствовать сложно структури-рованному формату документа;
  + случайный документ должен представлять собой текст, оформ-ленный 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 второгорода). Вероятность такого события является более высокой в сравнении с ве-роятностью наступления коллизии первого рода. В этом случае злоумыш-ленник «фабрикует» два документа (*M* и *M’*) с одинаковой ЭЦП и в нужный момент подменяет один документ другим. При использовании криптостой-кой хэш-функции такая атака должна быть также вычислительно сложной. Однако эти угрозы могут реализоваться из-за слабостей конкретных алго-ритмов хэширования, подписи или ошибок в их реализациях.

В заключение анализа атак на цифровые подписи укажем на сущест-вование 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*, кото-рые выполняют и иные функции.

***13.2 Сертификаты и сертификационные центры***

Использование надежных протоколов обмена ключами и защита за-крытого ключа от несанкционированного доступа позволяет снизить опас-ность упомянутых выше социальных атак.

153

Так как открытый ключ доступен любому пользователю, то необхо-дим механизм гарантии того, что этот ключ принадлежит именно указанному владельцу. Необходимо обеспечить доступ любого пользователя к подлин-ному 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 *серти-фикатов* или *сертификационных центров (СЦ)*,функции которых,в первомприближении, похожи на функции посредника, которые мы рассмотрели при анализе соответствующего алгоритма ЭЦП (см. п.9.2).

Структура сертификатов (или *сертификатов открытых ключей*) яв-ляется важной частью известного **протокола Х.509**. В соответствии с эти до-кументом основными элементами сертификата являются:

* данные о владельце и его открытый ключ;
* период действия сертификата;
* выдавшая сертификат организация.

Центр сертификации формирует закрытый ключ и собственный сер-тификат, формирует сертификаты конечных пользователей и удостоверяет их аутентичность своей цифровой подписью. Также центр проводит отзыв ис-текших и компрометированных сертификатов и ведет базы выданных и ото-званных сертификатов. Обратившись в СЦ, можно получить собственный сертификат открытого ключа, сертификат другого пользователя и узнать, ка-кие ключи отозваны.

* Республике Беларусь действуют несколько лицензированных СЦ (у нас они чаще называются удостоверяющими). К их числу относятся, напри-мер, 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 про-мышленности Республики Беларусь (http://cniitu.by/cert), Удостоверяющий центр национального центра маркетинга и конъюнктуры цен

(http://ca.ncmps.by/articles/view/8), Удостоверяющий центр Белорусской то-варной биржи (http://ecp.by/?page=280) и др.

Для получения сертификата между Центром и Клиентом заключается договор (соглашение). Формально сертификат оформляется в виде соответст-вующего документа (карточка открытого ключа). На рисунке 13.1 приведен фрагмент одного из таких документов.

Как мы ранее упоминали, значение открытого ключа хранится в за-щищенной базе данных, а программное обеспечение для генерации ЭЦП с тайным ключом передается Клиенту на носителе.

154

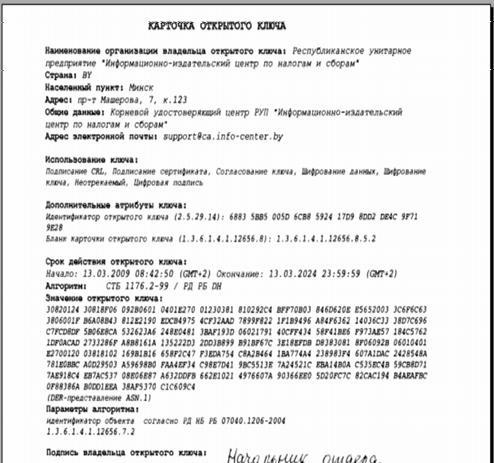


Рисунок 13.1. Пример оформления карточки открытого ключа

Закрытый ключ является наиболее уязвимым компонентом всей крип-тосистемы цифровой подписи. Злоумышленник, укравший закрытый ключ пользователя, может создать действительную цифровую подпись любого электронного документа от лица этого пользователя. Поэтому особое внима-ние нужно уделять способу хранения закрытого ключа.

* настоящее время используются следующие устройства хранения за-крытого ключа: смарт-карты, USB-брелки, таблетки Touch-Memory. Кража или потеря такого носителя может быть легко замечена пользователем, после чего соответствующий сертификат должен быть немедленно отозван. Из пе-речисленных устройств, вероятно, наибольшую степень защиты обеспечива-ет смарт-карта, поскольку ее использование требует ввода PIN-кода, то есть имеет место двухфакторная аутентификация. После успешного окончания процедуры аутентификации подписываемый документ или его хэш переда-ется в карту. Встроенный процессор осуществляет процедуры генерации ЭЦП. В процессе формирования подписи не происходит копирования закры-того ключа, поэтому все время существует только единственная его копия – на носителе.

155

***13.3 SSL-сертификаты***

Как известно, одним из сегментов информационного пространства, где проблема безопасного обмена сообщениями со взаимной идентификаци-ей участников процесса стоит особенно остро, является взаимодействие Кли-ента с *web-узлами* или *web-приложениями*. В наибольшей степени это касает-ся электронной торговли (или *Интернет-магазинов*), налоговой сферы и др.

* контексте рассматриваемой проблемы кратко охарактеризуем *биз-нес-модели* данной сферы деятельности.Основными из таких моделей счи-таются:

*бизнес—бизнес* (business-to-business, B2B);под этим обычно понима-

ется любая деятельность одних компаний по обеспечению других компаний сопроводительными услугами, дополнительным оборудованием, а также то-варами, предназначенными для производства других товаров; примером дея-тельности на основе этой модели может быть работа *аутсорсинговых* фирм-резидентов Парка высоких технологий (ПВТ) в РБ;

*бизнес—потребитель* (business-to-consumerилиbusiness-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);предполага-ет взаимодействие, например, между коммерческими и государственными структурами (вероятно, на основе этой модели осуществляется взаимодейст-вие между Администрацией ПВТ и фирмами-резидентами ПВТ;

*потребитель—администрация* (consumer-to-administration, C2A);примером может служить взаимодействие государственных структур и по-требителей в социальной и налоговой сфере (например, *электронное декла-рирование*).

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 защищѐнных сокетов).

SSL-сертификаты самый распространенный на данный момент тип сертификатов в Интернет. Из названия сертификата понятно, что он строится на основе *протокола* *SSL*. SSL изначально разработан компанией *Netscape* *Communications* для добавления протоколаhttpsв свой веб-браузерNetscapeNavigator. Впоследствии на основании протокола SSL 3.0 был разработан и принят *стандарт* *RFC*, получивший имя *TLS* (Transport Layer Security — безопасность транспортного уровня).

156

**Протокол SSL**.На рисунке13.2показано размещение данного прото-кола с 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).

*ННТР* *FTP* *SMTP*

*SSL*

*TCP*

*IP*

*Рисунок 13.2 Размещение SSL в стек протокола TCP/IP*

Архитектура и стек протоколов SSL представлены на рисунке 13.3.

|  |  |  |  |
| --- | --- | --- | --- |
| Протокол кви- | Протокол изменения | Протокол | HTTP |
| тирования SSL | параметров | извещения |  |
|  | шифрования SSL | SSL |  |
|  |  |  |  |

Протокол записи SSL

TCP

IP

Рисунок 13.4. Архитектура и стек протоколов SSL

*Протокол квитирования* отвечает за организацию *сеанса* (session)ме-жду Клиентом и Сервером. На его основе осуществляется взаимная аутенти-фикация сторон, согласовываются алгоритмы шифрования и криптографиче-ские ключи (на основе известного *алгоритма рукопожатия*).

*Протокол изменения параметров шифрования* генерирует однобайто-вое сообщение, которое дает указание начать копирование параметров со-

157

стояния ожидания в текущее состояние, что приводит к обновлению ком-плекта шифров, используемых для данного соединения.

*Протокол извещения* предназначен для передачи другой участвующейв обмене данными стороне извещений, касающихся работы SSL.

*Протокол записи* (это уровневый протокол)обеспечивает поддержкудвух сервисов для *соединения* (connection) между узлами системы: *конфиден-циальность* и *целостность сообщений*.Существует четыре протокола запи-си: *протокол рукопожатия* (handshake protocol), *протокол тревоги* (alert pro-tocol), *протокол изменения шифра* (the change cipher spec protocol), *протокол* *приложения* (application data protocol).

Для организации защищенного SSL-соединения на web-сервере необ-ходимо установить SSL-сертификат. SSL-сертификаты выдаются известными нам удостоверяющими или сертификационными центрами. Центров серти-фикации существует достаточно много, вот перечень самых популярных:

* Comodo — работает с 1998 г., штаб-квартира расположена в Нью-Джерси, США;
* Symantec — (бывший Verisign), в состав входит и Geotrust;
* Thawte — основан в 1995 г.;
* Trustwave — основан в 1995, штаб-квартира расположена в Чикаго, США. Самый крупный игрок на рынке SSL-сертификатов – это Symantec,

который владеет тремя крупнейшими центрами сертификации: Thawte, Ve-risgin и Geotrust.

* + РБ продажей SSL-сертификатов занимается ряд фирм. Например Хостер (https://hoster.by/service/services-and-solutions/ssl-certificates/), Экстме-диа (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 аутентифи-кации с помощью открытых ключей. Это комплексная система, которая свя-зывает открытые ключи с личностью пользователя посредством удостове-ряющего центра.

* основе PKI лежит использование криптографической системы с открытым ключом и несколько основных принципов:

158

* закрытый ключ известен только его владельцу;
* удостоверяющий центр создает сертификат открытого ключа, таким об-разом, 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 закры-тым ключом.

Таким образом, PKI представляет собой систему, основным компо-нентом которой является удостоверяющий центр и пользователи, взаимодей-ствующие 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 УЦ.

*Простая архитектура PKI*.Все пользователи доверяют одному УЦ,через который осуществляется взаимодействие между ними.

*Иерархическая архитектура PKI* –это наиболее часто встречающая-ся архитектура. Функционирует «под руководством» *Корневого* (Головного) УЦ. Частный пример иерархической PKI – корпоративная PKI.

*Сетевая архитектура PKI* –частный случай иерархической архи-тектуры. Но здесь нет Корневого центра. В этой архитектуре все УЦ дове-ряют рядом стоящим УЦ, а каждый пользователь доверяет только тому УЦ, который выдал SSL-сертификат.

*Архитектура кросс-сертифицированной корпоративной* PKIможетрассматриваться как гибрид иерархической и сетевой архитектур. Характери-зуется самой сложной цепочкой выдачи и анализа средств сертификации.

*Архитектура мостового УЦ* разрабатывалась для сглаживания недос-татки предыдущей архитектуры. В данном случае все компании (владельцы сертификатов) «доверяют» одному, *мостовому УЦ*, который является прак-тически их головным УЦ, но также выступает в роли посредника между дру-гими 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 перечислен-ных выше центров сертификации, то их корневые сертификаты установлены практически во всех существующих браузерах

159

Чтобы проверить, корневые сертификаты каких центров сертифика-ции установлены в вашем браузере, достаточно в настройках браузера найти такую опцию30.

* настоящее время сертификационные центры используют ключи длиной 2048 бит на основе RSA.

*Вопросы для контроля и самоконтроля*

1. Классифицировать атаки на ЭЦП.
2. Охарактеризовать коллизии, которые могут возникнуть при исполь-зовании ЭЦП.
3. Что такое карточка открытого ключа?
4. Охарактеризовать основные бизнес-модели.
5. Для чего создаются и как применяются SSLсертификаты?
6. Охарактеризовать структуру протокола SSL.
7. Дать сравнительную характеристику архитектур PKI.
8. Воспользовавшись кнопкой «Office» (находится в верхнем левом углу соответствующего приложения), добавьте цифровую подпись к доку-менту, например, MS Word.
9. Полезную информацию о практическом использовании SSL-сертификатов и о СЦ можно найти на странице http://habrahabr.ru/company/tuthost/blog/150433/

160

**ГЛАВА V**

**МЕТОДЫ ОБФУСКАЦИИ В ЗАЩИТЕ ИНФОРМАЦИИ**

**14 Основные методы защиты программного кода**

***14.1 Общая сравнительная характеристика методов защиты ПО***

Появляющиеся в доступных источниках информации сведения об атаках на продукты лидеров производства программного обеспечения (ПО) – наглядное доказательство того, что на сегодняшний день еще нет действи-тельно эффективной защиты ПО. У этого есть ряд причин.

* 1. *Сложность* –использование сторонних компонентов,петель об-ратной связи, многовариантность реализации, взаимодействие с другими приложениями, неконтролируемые каналы передачи данных – учесть все это очень сложно.
  2. *Баги* –уникальный вид ошибок программирования,которые воз-никают при плохой реализации того или иного решения, но, в отличие от ошибок, не мешают выполнению основной логики программы, а лишь сни-жают 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 про-грамм, например, операционной система Windows XP насчитывает около 40 млн. строк кода.
  3. *Удобство атак*.Можно проводить атаки на расстоянии,организо-вывать распределенные атаки, подбирать ключи с помощью мощных вычис-лительных систем, использовать одни и те же приемы взлома для схожих ви-дов ПО, использовать известные виды атак, пользуясь тем, что большинство пользователей крайне медлительны в установке обновлений.
  4. *«Бесплатность» воспроизводства*.Специфика хранения про-

граммных продуктов делает стоимость их копирования практически нулевой.

* 1. *Популярность*.В современном мире практически ни одна сферажизнедеятельности 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 ве-роятности ;

161

* ограничения возможностей использования (чем надежней защита, тем меньше функций может выполнять ПО);
* повышенные требования к соблюдению правил информационной безо-пасности 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 ресур-сы (программные, аппаратные, временн****ы****е), и другие*.

Одним из направлений защиты ПО является *защита кода*, поскольку его анализ является одной из ключевых составляющих при взломе. Среди ме-тодов защиты можно выделить следующие:

* использование *цифровых водяных знаков* (software watermark), основы-

вающееся на записи в код программы скрытой информации (кому принадле-жат 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 стеганографиче-ских методов;

* *отпечатки пальца* (software fingerprint)–технология,которая,кроме за-писи информации, позволяющей доказать право собственности на програм-му, требует записи в каждую копию программы уникального идентификаци-онного кода, присваиваемого каждому покупателю программы, что позволя-ет впоследствии отследить нарушителя авторского права;
* *определение подлинности кода* (tamper-proofing)–в код программы по-мещается *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 вне-сены изменения, она делает программу нефункциональной; упомянутые про-цедуры могут строится, например, на основе известных нам методов хеши-рования или кодов четности (CRC);
* *зашифровывание кода программы*,после чего она в зашифрованном видепоставляется конечным пользователям (иногда эффективно зашифровывать только наиболее важные, критические, участки кода, а не весь код програм-мы); когда пользователь запускает такую программу, вначале будет иниции-рована процедура ее расшифрования (с помощью некоторого ключа).

Недостатки перечисленных методов могут состоять в том, что у зло-умышленника может появиться возможность подвергнуть изменению водя-ной знак или отпечаток пальца, либо после приобретения лицензионной ко-пии программы произвести извлечение расшифрованных частей программы в процессе ее работы из памяти.

* + большинстве случаев для обхода защиты взломщику требуется изу-чить принцип ее работы, то, как она взаимодействует с самой защищаемой программой. Этот процесс называется процессом *реверсивной* *(обратной)*

*инженерии* или *реинжинирингом*.

162

* последнее время специалисты считают, что наиболее эффективным средством противодействия несанкционированному использованию или ре-инжинирингу ПО является *обфускация программного кода*. Именно этот класс методов и будет являться предметом нашего дальнейшего анализа.

***14.2 Характеристика*** [***современных методов обфускации кода***](https://docs.google.com/document/d/1rFFwjOv0QF_ztI5mDmmb4EJcoEYV1WYRGJ-B1s9Sd68/edit#heading=h.pe14fimlal31)

**Определение 14.1. *Обфускация* (от англ. obfuscate – делать непо-нятным, запутывать) – внесение в исходный код программы таких мо-дификаций, которые не изменяют поведение программы, ее функцио-нальность (заложенный алгоритм), но значительно затрудняют понима-ние указанного алгоритма.**

*Формальное определение задачи обфускации: придумать алгоритм, который, получая на вход некоторую программу P, преобразует ее в про-грамму O(P) таким образом, что код программы O(P) не должен раскры-вать информацию об алгоритмах, параметрах, и внутренней структуре этой программы P.*

* + связи с этим в основу методов обфускации положены следующие принципы:
* добавления 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 чувстви-телен к пробельным знакам.

При разработке методов обфускации необходимо соблюдать требова-

ния:

* 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 затрата-ми на его применение.

163

***14.2.1 Уровни обфускации***

Программный код может находиться в нескольких формах: исходной, промежуточной, в машинный код.

*Обфускация на уровне исходного кода*.Данный тип обфускации можетиспользоваться, если необходимо защитить программу на этапе ее разработ-ки (например, от недобросовестных офисных сотрудников); если программа распространяется в виде исходных кодов (клиентские скрипты языков про-граммирования для Интернета, например, JavaScript, VBScriptи др.); если это серверные библиотеки или *CMS-системы*, предоставляемые заказчику с ог-раниченными возможностями использования или распространения.

На данном уровне исходный и защищенный код, *P* и O(*P*), представ-лены одним языком программирования. Преимуществом создания алгорит-мов 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 оп-тимизации современных компиляторов многие неиспользуемые участки кода просто не входят в сборку (в том числе – и специальный «мусор», создавае-мый некоторыми алгоритмами). Соответственно, специалисты, изучающие данный код на более низком уровне, не столкнутся со многими препятствия-ми. Дополнительные ограничения возникают из-за строгости семантики кода исходного уровня, не позволяющей приводить код к виду, содержащему не-читаемые символы в названиях объектов, использовать смежную область па-мяти и др.

*Обфускация на промежуточном уровне.* Особый вид программногокода появился благодаря технологии выполнения ПО на виртуальных маши-нах (NET, Java, Flash). Данный код является более сложным для понимания, чем исходный, т.к. переводит его команды в более универсальный набор. Тем не менее, оперировать этими командами гораздо легче, чем дизассемблиро-ванным кодом. Для таких команд созданы специальные декомпиляторы, по-зволяющие получить на их основе результат близкий или даже идентичный (в случае с Flash) исходному коду. Большинство новых программных про-дуктов разрабатывается именно для работы на виртуальных машинах. Про-блема защиты кода данного уровня стоит наиболее остро***.***

*Обфускация на машинном уровне*.Является наиболее сложным меха-низмом 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 рас-путывания алгоритмы обфускации, которые могут не только затруднить изу-чение кода, но и вообще помешать получить данные для изучения с помо-щью каких бы то ни было инструментов реинжиниринга.

164

*Особенности обфускации сетевых приложений*.Применение обфус-кации к интернет-приложениям имеет ряд особенностей, поскольку такие приложения обладают свойствами, не характерными для настольного про-граммного обеспечения (ПО), а именно:

* массовый доступ – популярные Интернет-ресурсы в минуту обслужива-ют 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 клиент-ской части приложения, но и серверной, которая выдается заказчику с огра-ниченными 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 ре-сурсов для выполнения программы, т.к. производит запутывание за счет вне-дрение дополнительного кода (добавляет лишнее ветвление, циклы, пере-менные и т.п.). Но есть и такие приемы, которые в результате не только за-трудняют изучение кода, но и делают его выполнение менее затратным. К их числу относятся:

* *минификация* –за счет удаления комментариев и необязательных пробель-ных 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 раз-личных 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 статическихданных для последующего использования; актуально для динамических ме-тодов обфускации, которые выполняются каждый раз при обращении к про-грамме; временное или частичное сохранение результатов их работы позво-ляет значительно снизить нагрузку на сервер, что является критичным для многих масштабных проектов.

165

***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 использо-вать перехваченные исключения аналогично меткам перехода в операторе *goto*.Данная технология активно используется при обфускацииSkype31.

*Пример 14-1*.Например,простейшую проверку булевой переменной:

 if(flag)



code1

 else



code2

можно преобразовать к следующему варианту:

 try:



x = 1 / (1 - flag)



code2

 except ZeroDivisionError:



code1

*Пример 14-2*.Учитывая возможность фильтрации исключения по ти-пу, 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 еѐ необходимость.

1. Интернет-портал [Электронный ресурс] / Википедия. – Режим доступа: [http://ru.wikipedia.org/wiki/CLEFIA.](http://ru.wikipedia.org/wiki/CLEFIA) – Дата доступа: 15.04.2014

166

*Перемешивание*.Идея заключается в разделении кода на блоки,рас-положенные в псевдослучайном порядке внутри оператора switch. Сам опе-ратор switch находится внутри цикла, который срабатывает необходимое ко-личество 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 *case*-ы» и пустые действия внутри блоков. Также можно реализовать менее оче-видный механизм присвоения значений переменной *step*, в том числе – и раз-новариантный (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:

167

* *проблема останова* –не существует общего алгоритма,позволяющего оп-ределить завершенность заданных действий при известных входных значе-ниях, по описанию алгоритма и входным данным;
* «*десятая проблема Гильберта*»32 – неразрешимость диофановых уравне-ний общего вида за конечное число операций при произвольных неизвестных и целых рациональных числовых коэффициентах; например, следующее диофаново уравнение не имеет решений в целых рациональных числах:

*(x+1)\*(x+1)\*(x+1) + (y+1)\*(y+1)\*(y+1) - (z+1)\*(z+1)\*(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 Коллатца33,на-оборот: 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 гене-рируемых 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) | & |

1. состоит в нахождении универсального метода целочисленного решения произвольно-

го алгебраического диофантова уравнения; см.: [Матиясевич,](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B8%D1%8F%D1%81%D0%B5%D0%B2%D0%B8%D1%87,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B8%D1%87) Ю.В. [Десятая проблема Гильберта](http://bookfi.org/book/448136)[/Ю. В.](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B8%D1%8F%D1%81%D0%B5%D0%B2%D0%B8%D1%87,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B8%D1%87)

[Матиясевич.](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B8%D1%8F%D1%81%D0%B5%D0%B2%D0%B8%D1%87,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B8%D1%87) — М.: Наука. – 1993. – 128 с

1. См.: Стюарт, И. Величайшие математические задачи/ И. Стюарт. — М.: Альпина нон-фикшн,

2015. — 460 с.

168

 (!x2 | !x1 | !x3) & (!x4 | !x0 | !x1) & (!x4 | !x0 | !x3) & (!x4 |  !x1 | !x3) & (!x0 | !x1 | !x3)

***14.2.2.3 Субституция***

Идея основана на обращении к полям и методам объекта, как к клю-чам ассоциативного массива.

Рассмотрим пример.

*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 сущно-сти такого подхода мы кратко вспоминали в начале данной главы).

***14.2.2.4 Кодирование строковых значений***

Различное изменение «внешнего вида» кода, а также строковых зна-чений позволяет затруднить использование прямого поиска для анализа кода. Рассмотрим некоторые особенности такого подхода.

***Шестнадцатеричная кодировка****.*Проиллюстрируем это примером.*Пример 14-8*.Наиболее простым является кодирование строки в шест-

надцатеричный 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"] = "copy-  right";

После кодирования последний фрагмент кода может принять сле-дующий вид:

 this["х\64х\6fх\63х\75х\6dх\65х\6eх\74"]["х\67х\65х\74х\45х\6cх\65х

 \6dх\65х\6eх\74х\42х\79х\49х\64"]("х\73х\70х\65х\63х\2dх\69х\64")["

 х\69х\6eх\6eх\65х\72х\48х\54х\4dх\4c"] =

 "х\63х\6fх\70х\79х\72х\69х\67х\68х\74";

169

***Генерация данных****.*Более сложные преобразования можно осуществ-лять с помощью различных операций, приводящих к появлению строковых результатов. Например, чтобы сгенерировать строку "getElementById", мож-но 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 = "getElementBy-

 Id"

 }

Другой способ получения символов – манипулирование числовыми операциями. С учетом того, что некоторые объекты могут неявно преобразо-вываться в числовые значения (пустой массив – 0, true – 1 и т.д.), можно по-лучать произвольные числовые значения, не используя ни одного числового символа.

*Пример 14-10*.Чтобы получить число123,можно применить следующую по-следовательность 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* бу-дет преобразован обфускатором-кодировщиком *jjencode* к следующему виду:

 $=~[];$={\_\_\_:++$,$$$$:(![]+"")[$],\_\_$:++$,$\_$\_:(![]+"")[$],\_$\_:++$,$ \_$$:({}+"")[$],$$\_$:($[$]+"")[$],\_$$:++$,$$$\_:(!""+"")[$],$\_\_:++$,$\_



 $:++$,$$\_\_:({}+"")[$],$$\_:++$,$$$:++$,$\_\_\_:++$,$\_\_$:++$};$.$\_=($.$\_=

 $+"")[$.$\_$]+($.\_$=$.$\_[$.\_\_$])+($.$$=($.$+"")[$.\_\_$])+((!$)+"")[$.\_

170

 $$]+($.\_\_=$.$\_[$.$$\_])+($.$=(!""+"")[$.\_\_$])+($.\_=(!""+"")[$.\_$\_])+$

 .$\_[$.$\_$]+$.\_\_+$.\_$+$.$;$.$$=$.$+(!""+"")[$.\_$$]+$.\_\_+$.\_+$.$+$.$$;



$.$=($.\_\_\_)[$.$\_][$.$\_];$.$($.$($.$$+"\""+$.$\_$\_+(![]+"")[$.\_$\_]+$.$

 $$\_+"\\"+$.\_\_$+$.$$\_+$.\_$\_+$.\_\_+"("+$.\_$\_+"\*"+$.\_$\_+");"+"\"")())();

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)) ('\_');

* 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");

171

***Кодирование невидимыми символами****.*Предусматривает запись ко-дового значения символов с помощью невидимых знаков (табуляций, пробе-лов, 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

* последнем случае в качестве разделения используется перенос строки, благодаря чему можно не соблюдать одинаковую длину записи сим-волов. Это выгодно при использовании большого количества символов, на-ходящихся в начале кодировочной таблицы. Иначе будет экономней исполь-зовать равную длину, задействовав при этом возврат каретки и перенос стро-ки как в качестве дополнительных символов для кодированию, получив та-ким образом систему счисления по основанию 4 и уменьшив, тем самым, на единицу порядок размерность длины.

***14.2.3 Использование других форматов хранения***

Усложнить анализ кода можно, разместив его фрагменты за предела-ми 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>

172

* результате выполнения данного скрипта в оператор *eval* будет пере-дана строка *alert(1)*, полученная в результате представления содержимого *body*,как строки с последующем ее разделением по атрибутам *a*, *b*, *c* и извле-чением правых частей, из которых затем извлекается содержимое между ка-вычками аналогичным способом.

***Упаковка в изображение***.Технология хранения кода в виде изобра-жения позволяет реализовывать его асинхронную загрузку, скрытие от мони-торинга загружаемых javascript-файлов, а также сжатие кода.

*Пример 14-15*.Пример того,как выглядит реализованная в виде изо-бражения библиотека *jquery,* представлен на рисунке 14.1

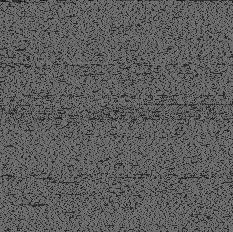


Рисунок 14.1 – Код библиотеки *jquery*, упакованный в изображение

Для восстановления кода пиксели изображения преобразуются в чи-словые 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**

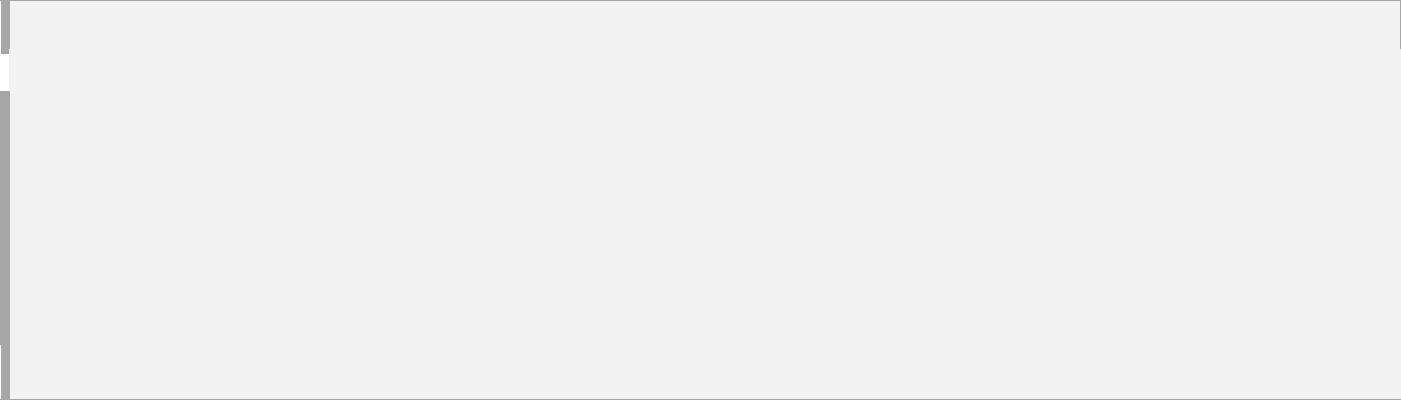
***14.3.1 Использование интерпретируемых языков***

***программирования***

173

Метод является серьезной модификацией известного *метода преобра-зования кода в последовательность операторов*.При этом новый метод сво-боден от основных недостатков, свойственных известному. Покажем это на примерах.

*Пример 14-16. Статическое расположение операторов*,выполняющихкод, например, может выглядеть так:



 $=~[];$={\_\_\_:++$,$$$$:(![]+"")[$],\_\_$:++$,$\_$\_:(![]+"")[$],\_$\_:++$,$\_

$$:({}+"")[$],$$\_$:($[$]+"")[$],\_$$:++$,$$$\_:(!""+"")[$],$\_\_:++$,$\_

$:++$,$$\_\_:({}+"")[$],$$\_:++$,$$$:++$,$\_\_\_:++$,$\_\_$:++$};$.$\_=($.$\_

=$+"")[$.$\_$]+($.\_$=$.$\_[$.\_\_$])+($.$$=($.$+"")[$.\_\_$])+((!$)+"")[$

.\_$$]+($.\_\_=$.$\_[$.$$\_])+($.$=(!""+"")[$.\_\_$])+($.\_=(!""+"")[$.\_$\_]

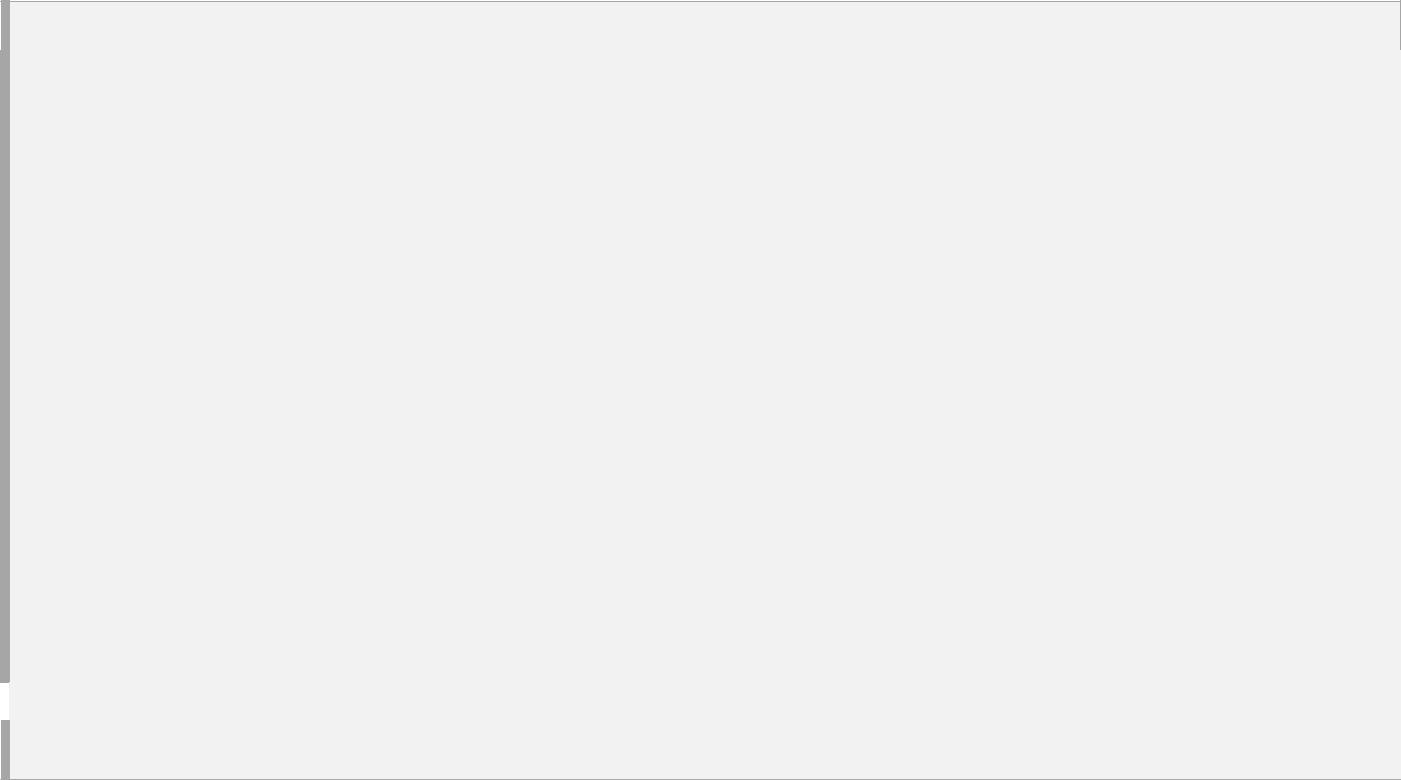
)+$.$\_[$.$\_$]+$.\_\_+$.\_$+$.$;$.$$=$.$+(!""+"")[$.\_$$]+$.\_\_+$.\_+$.$+$

.$$;$.$=($.\_\_\_)[$.$\_][$.$\_];$.$($.$($.$$+"\""+**!CODE!**+"\"")())();

Данный подход к записи операторов не зависит от защищаемого кода

* располагается в конструкции всегда одним и тем же образом. Вследствие чего 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 протяжениивсего выполнения кода, хранение всех значений в одном массиве:



$ = {

\_\_\_: ++$, // 0

$$$$: (![] + "")[$], // f

\_\_$: ++$, // 1

$\_$\_: (![] + "")[$], // a

\_$\_: ++$, //2

$\_$$: ({} + "")[$], // b

$$\_$: ($[$] + "")[$], // d

\_$$: ++$, // 3

$$$\_: (!"" + "")[$], // e

$\_\_: ++$, // 4

$\_$: ++$, // 5

$$\_\_: ({} + "")[$], // c

$$\_: ++$, // 6

$$$: ++$, // 7

$\_\_\_: ++$, // 8

$\_\_$: ++$ // 9

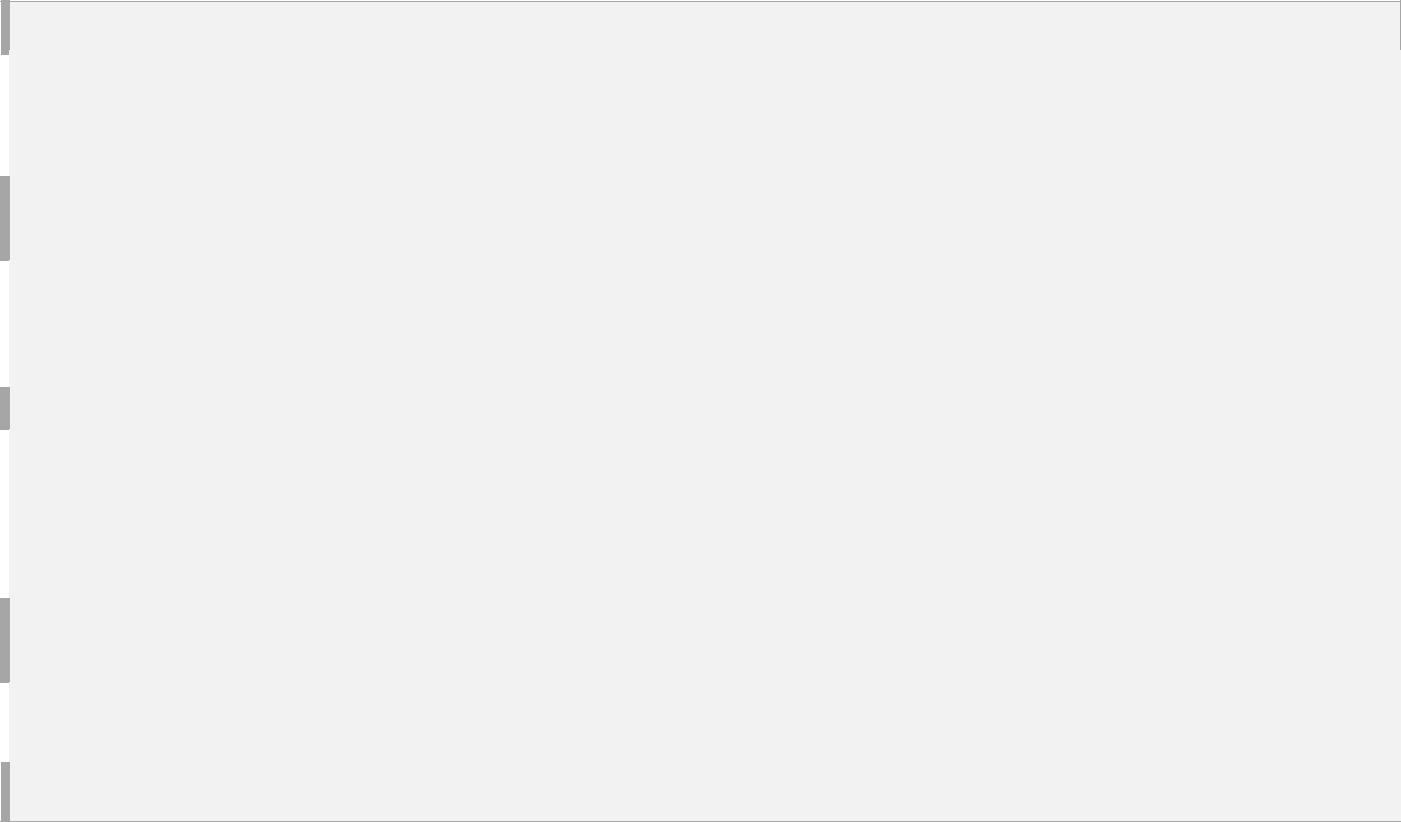
};

Подобный подход подвержен обратному декодированию, основанно-му на последовательном обходе ключей соответствующего массива.

*Пример 14-18. Преобразование символов в операторы по принципу «один к одному»*,простые приемы получения дополнительных значений засчет использования индекса строки и несложных для проверки инструкций,

174

деобфускацию которых можно провести путем комбинаций двух замен, ос-нованных 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) {}

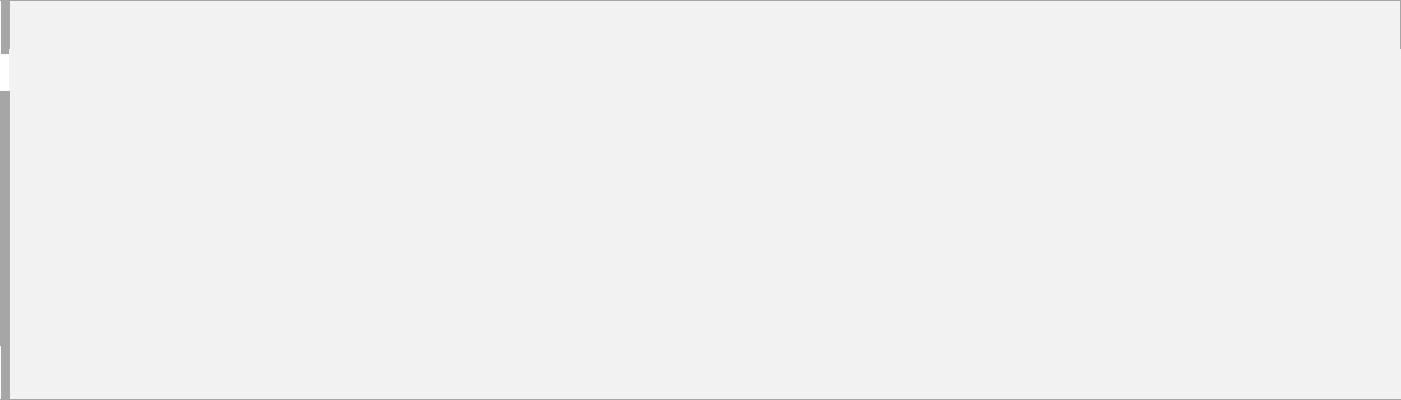
}

Для устранения выявленных и продемонстрированных на примерах уязвимостей были использованы следующие идеи.

1. ***Разбиение исполняемого кода на несколько частей***,перемешан-

ных методом *ChenxiWang`s*, как это показано в двух нижеследующих приме-рах.

*Пример 14-19.*



intf(){

 intx, y, z;

x=0; //блок1

y=1; // блок2

z=2; // блок 3

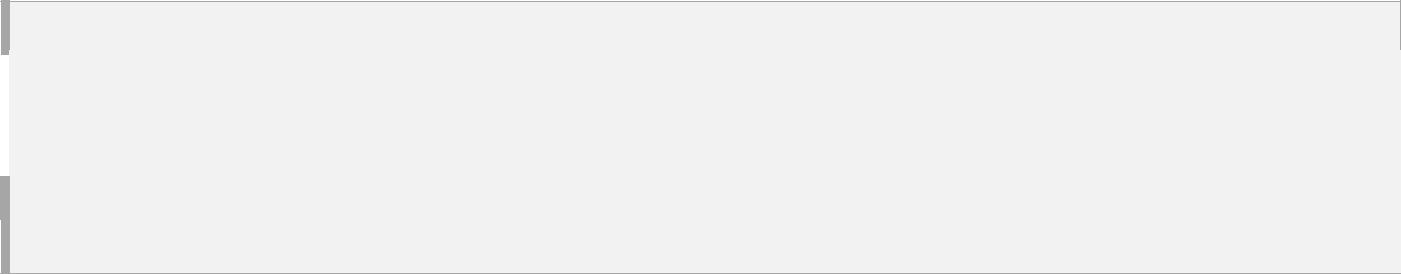
if(y<x) z=0; // блок4

y=-1; // блок 5

if(y>x) z=1 // блок 6

return z;

*Пример 14-20.*



intf(){

 intx, y, z;

 int step=6;

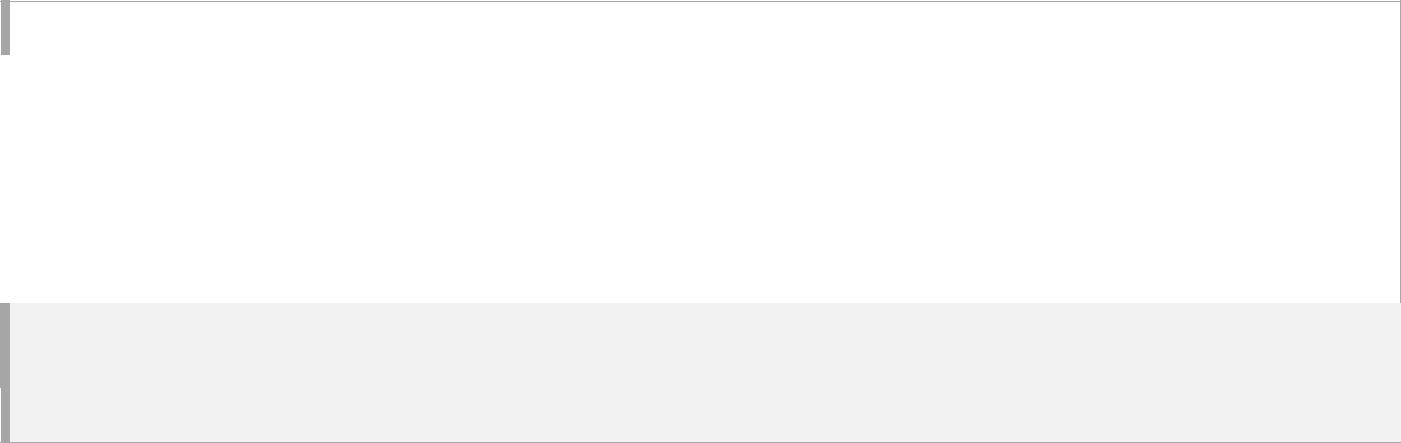
 while(step)

{

switch(step)

175

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | { |  |  |  |  |  |
|  |  | 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 | |

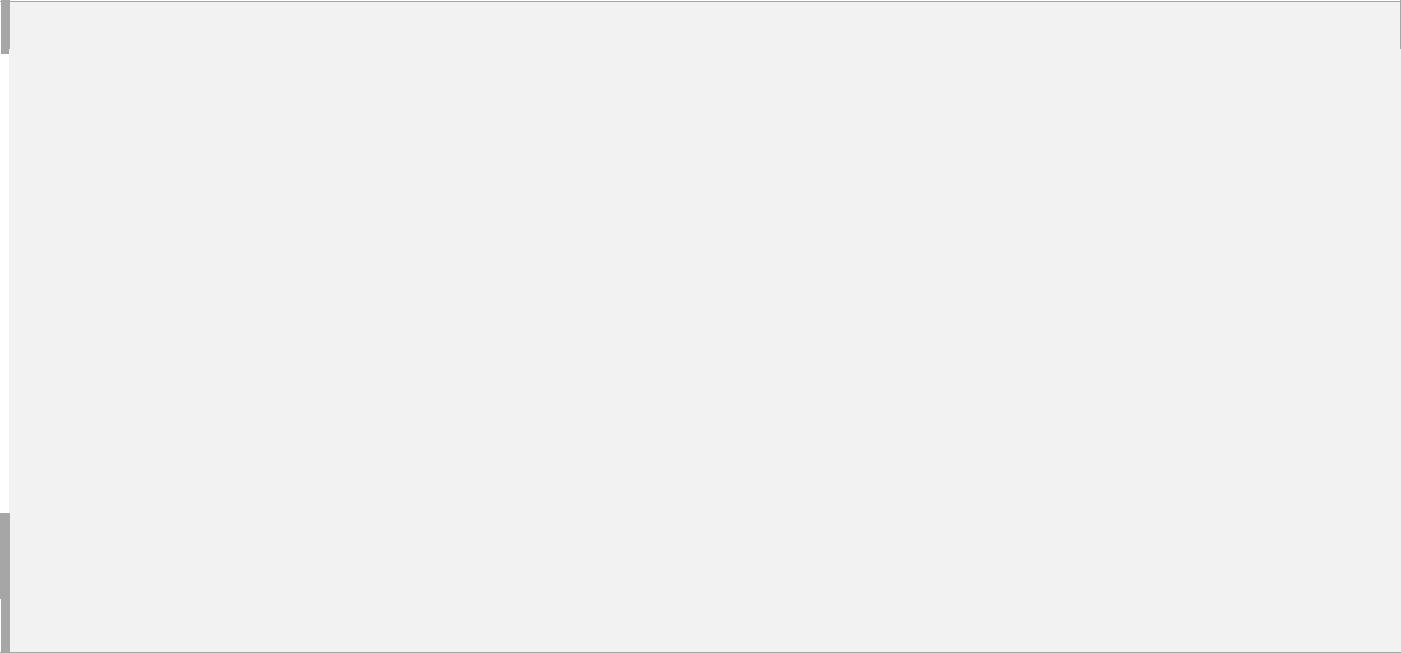
default step = 0;

}

return z;}

*2.****Генерация строковых значений в отдельных функциях***,что реали-зовано 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;

 }

 }

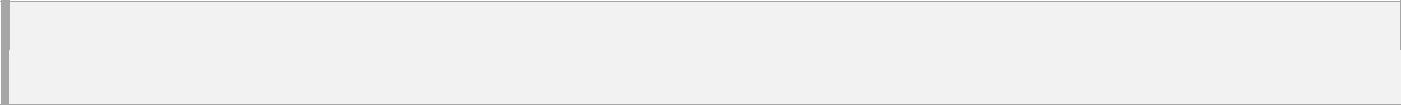
 return str;

}

string str = genStr([8,16,4,4,133]); // Hello

1. ***Использование тождественных либо допустимых по точности математических операций*** для записи чисел.

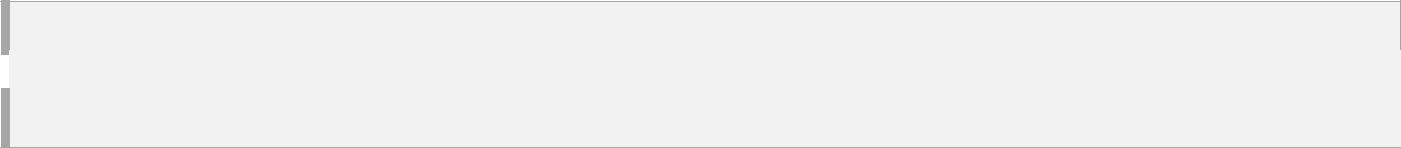
*Пример 14-22*.



y = DataTime.Now;

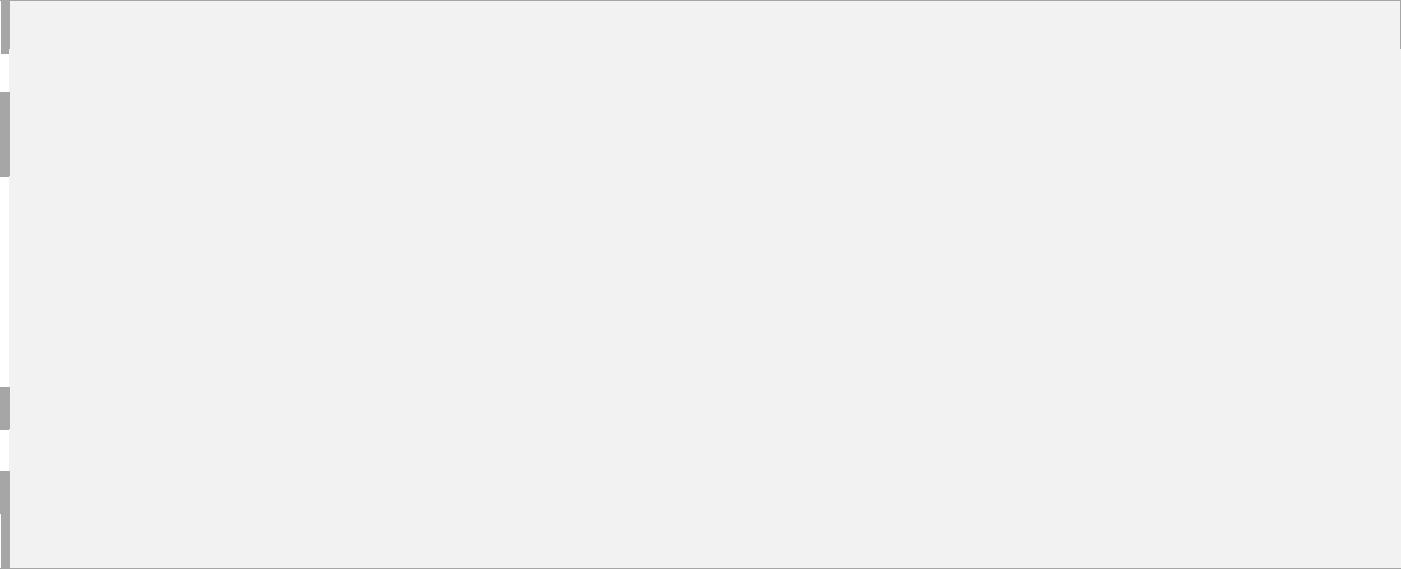
1. = у%10+((у%100)/10)%10)-(у%100)%9+1;// x = 1;
   1. ***Использование дополнительных приемов кодирования***,основанныхна субституции строк и генерации исключений.

*Пример 14-23*.Пример показывает кодирование методов с помощью *строковых обѐрток и исключений*:



document.getElementsByTagName("html")[0].innerHTML =  document.getElementsByTagName("body")[0].length;

176

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***

*Сущность метода*.Поскольку поиск и изменение строковых данных вобфусцированом коде является одним из ключевых, а порой и основных мо-ментов атаки, в целях большей степени защиты можно использовать шифро-вание кода на на ключе, задаваемом в виде *рекуррентного математического* *соотношения*.

Использование рекуррентных математических соотношений позволя-ет 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 объ-емом данных и часто повторяющимися последовательностями символов.

Итак*,* *сущность* предлагаемого подхода состоит в следующем. К не-которому однозначному числовому определению шифруемого символа *A* (например, кодовому значению из используемой таблицы) добавляется неко-торое 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 со-общении.

Спецификой математической формулы является возможность получе-ния одинаковых значений для ряда передаваемых переменных при различ-ных наборах математических операций. Благодаря этому нельзя с уверенно-стью полагаться на то, что ключ найден, если с его помощью удалось рас-шифровать отдельный набор данных.

Но использование простых математических операций не представляет значительной сложности для операции криптоанализа (слома шифра). По-этому важнейшей особенностью данного подхода являются динамические и рекуррентные преобразования. В качестве примера динамических преобразо-ваний можно привести операцию деления по модулю, изменяемому в про-

177

цессе использования по заданному правилу, в качестве рекуррентного преоб-разования – 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 час-тях выражения. Также добавлена возможность использования операции ин-крементации.

Существует возможность составления формулы, специально подоб-ранной для скрываемого текста, благодаря чему можно привести его к после-довательности одинаковых символов.

*Пример 14-24*.Пусть дан текст«*абвгд*».Если его зашифровать с по-мощью 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)*.

Стохастический разброс получаемых значений в результате примене-ние различных функций приведен в таблице 14.1.

Таблица 14.1 – Сдвиг данных при различных математических соотношениях

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Формула** | **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 соотношений по сравне-нию 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:

178

* зависимость криптостойкости ключа от составленной формулы, что огра-ничивает 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 математиче-ские 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 последовательностьнечитабельных 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 тождест-венных переменных, с их произвольным выбором на соответствующих фраг-ментах кода;
* изменение значений переменных на протяжении всего кода, что не позво-ляет производить обратное преобразование на основе прямолинейной заме-ны.

Реализована модель составления и хранения обфусцирующих конст-рукций 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. (см. рисунок 14.2).

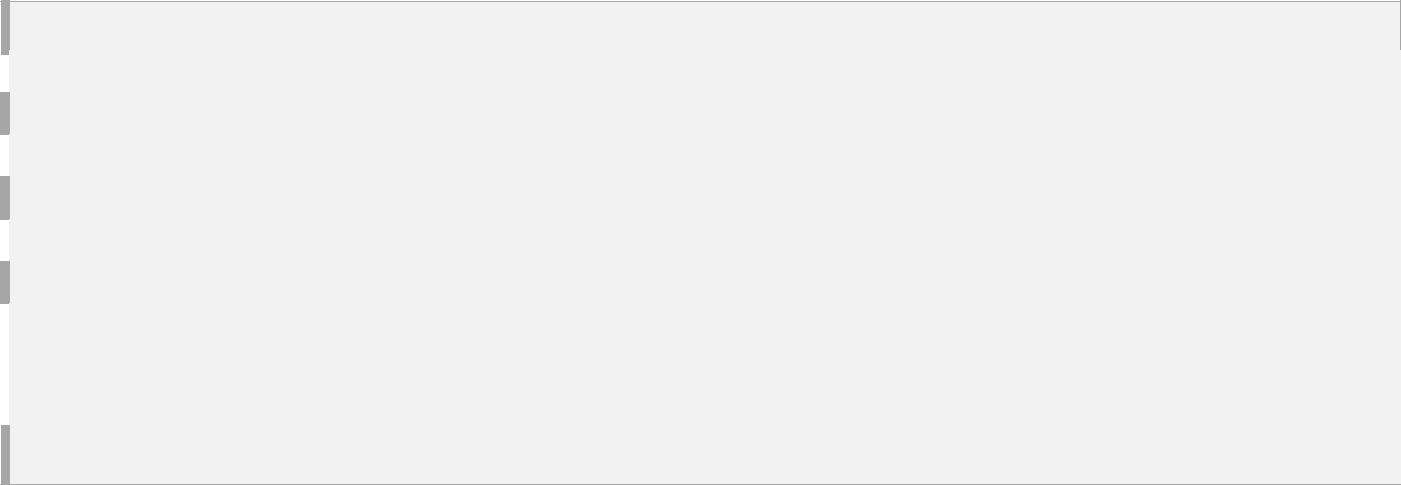
* + качестве источника для заполнения разработанной базы использо-вался ресурс *wtfjs*, из которого были извлечены 95 неочевидных по выполне-нию приѐма, среди которых есть такие как: работа методов класса *Math* при передаче разнотипных данных; операции над массивами и коллекциями; ма-нипулирование областями 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.

179



Рисунок 14.2 – Общая модель хранения шаблонов кода, используемого при генерации обфусцирующих вставок

*Пример 14-26*.Пример конструкции для манипулирования областямивидимости объектов выглядит так:



 var **a** = 1, // используется запятая

**b** = 1;

 (function(){

var **a** = 2 // НЕ используется запятая

 **b** = 2; //в результате не создается локальнаяb,а используется

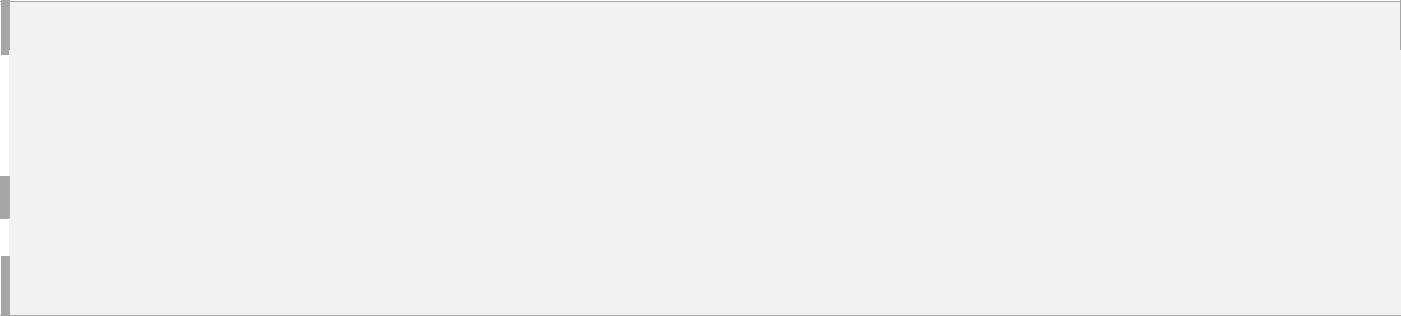
внешняя

 }());

 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]]

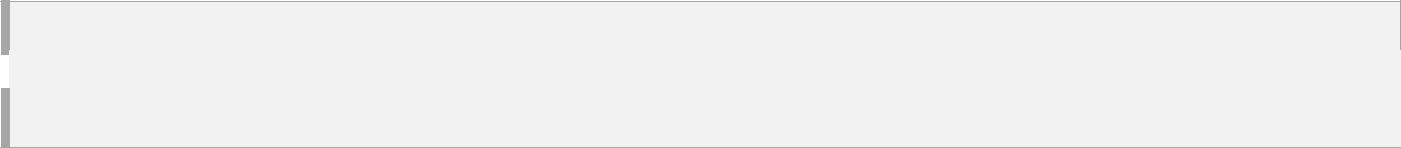
180

}());

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 преобразо-вано 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 ра-нее при описании *тождественных либо допустимых по точности матема-тических операций* для записи чисел,преобразуется к следующему шаблону(хранение заменителя, основанного на математическом тождестве):



[[NAME2]] = DataTime.Now;

 [[NAME1]] = [[NAME2]]%10+(([[NAME2]]%100)/10)%10)-([[NAME2]]%100)%9+[[VALUE1]];

* этом шаблоне используется дополнительная переменная *NAME2*. Если ее название не будет установлено, то используется значение по умолча-нию.

Операции обфускации являются необратимыми. Тем не менее, было бы полезным иметь возможность восстановить после их применения исход-ный код, сохранив для этого в отдельном файле необходимые данные (клю-чи). Так же полезным представляется использования регулярных выражений для поиска и замены по собственным правилам.

*Вопросы для контроля и самоконтроля*

1. Дать сравнительную оценку и пояснить сущность известных мето-дов защиты кодов программ от несанкционированного использования и мо-дификации.
2. На чем основана идея методов обфускации?
3. Продемонстрировать на конкретном примере реализацию выбран-ного (или указанного преподавателем) метода обфускации программного ко-да.

181

**ОГЛАВЛЕНИЕ**

|  |  |
| --- | --- |
| **Введение** | 3 |

**ГЛАВА I МЕТОДЫ ЗАЩИТЫ ИНФОРМА-ЦИИ И ИНФОРМАЦИОННАЯ БЕЗОПАС-НОСТЬ СИСТЕМ**

1. **Фундаментальные понятия и определения из области ин-**

|  |  |  |
| --- | --- | --- |
|  | **формационной безопасности систем** | 5 |
| 1.1 | Краткая историческая справка | 5 |
| 1.2 | Основные понятия и определения из области |  |
|  | защиты информации | 7 |
| 1.3 | Общая характеристика факторов, влияющих на безопасность и |  |
|  | надежность ИВС | 9 |

1. **Потенциальные угрозы безопасности информации в ИВС.**

|  |  |  |
| --- | --- | --- |
|  | **Объекты и методы защиты информации** | 11 |
| 2.1 | Естественные и искусственные помехи и угрозы безопасности | 11 |
| 2.2 | Основные методы и средства повышения безопасности ИС и |  |
|  | ИВС | 17 |

**ГЛАВА II КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ**

1. **Сущность криптографического преобразования информа-**

|  |  |  |
| --- | --- | --- |
|  | **ции** | 30 |
| 3.1 | Основы понятия предметной области. Цели и задачи крипто- |  |
|  | графии | 30 |
| 3.2 | Подстановочные и перестановочные шифры | 32 |
| 3.2.1 | *Подстановочные шифры* | 32 |
| 3.2.2 | *Перестановочные шифры* | 37 |
| 3.3 | Симметричные и асимметричные шифры | 38 |
| 3.4 | Блочные и потоковые шифры | 39 |
| 3.5 | Особенности криптоанализа | 39 |
| **4** | **Основы криптографии** | 43 |
| 4.1 | Элементы теории чисел | 43 |
| 4.2 | Основы модулярной арифметики | 46 |
| 4.3 | Обратные значения чисел в модулярной арифметике | 47 |
| 4.4 | Проблема дискретного логарифма | 50 |

1. **Характеристики и реализация криптографических алго-**

|  |  |  |
| --- | --- | --- |
|  | **ритмов** | 51 |
| 5.1 | Алгоритм DES | 51 |
| 5.2 | Модификации алгоритма DES | 57 |
| 5.3 | Другие блочные алгоритмы | 58 |
| 5.4 | Асимметричная криптография | 60 |
| 5.4.1 | *Основы асимметричной криптографии* | 60 |
| 5.4.2 | *Алгоритм RSA* | 62 |
|  |  | 182 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5.4.3 | *Алгоритм Эль-Гамаля* | 65 |  |
| 5.4.4 | *Криптосистемы на эллиптических кривых* | 68 |  |
| 5.4.4.1 | *Представление и описание эллиптической кривой* | 68 |  |
|  |  |  |
| 5.4.4.2 | *Рекомендации по выбору параметров эллиптической кривой* | 77 |  |
| 5.4.4.3 | *Система распределения криптографических ключей на осно-* |  |  |
|  | *ве эллиптической кривой* | 78 |  |
| **6** | **Потоковые шифры** | 81 |  |
| 6.1 | Общая характеристика потоковых шифров | 81 |  |
| 6.2 | Синхронные потоковые шифры | 83 |  |
| 6.3 | Самосинхронизирующиеся потоковые шифры | 84 |  |
| 6.4 | Генераторы псевдослучайных последовательностей для |  |  |
|  | потоковых шифров | 86 |  |
| 6.4.1 | *Линейные конгруэнтные генераторы* | 86 |  |
| 6.4.2 | *Генераторы ПСП на основе регистров сдвига* |  |  |
|  | *с линейной обратной связью* | 87 |  |

**ГЛАВА III СТЕГАНОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ**

1. **Структура, особенности построения и использования**

|  |  |  |
| --- | --- | --- |
|  | **стеганографических систем** | 91 |
| 7.1 | Терминология, сущность и цели стеганографического преоб- |  |
|  | разования информации | 91 |
| 7.2 | Основные направления использования стеганографических |  |
|  | систем | 97 |
| 7.2.1 | *Методы текстовой стеганографии* | 99 |
| 7.2.1.1 | *Синтаксические методы текстовой стеганографии* | 99 |
| 7.2.1.2 | *Лингвистические методы текстовой стеганографии* | 104 |
| 7.2.2 | *Стеганографические методы на основы избыточности среды* | 108 |
| 7.3 | Основные принципы стеганографического анализа | 111 |
| **8** | **Моделирование стеганографической системы** | 115 |
|  | **ГЛАВА IV ЭЛЕКТРОННАЯ ЦИФРОВАЯ** |  |
|  | **ПОДПИСЬ** |  |
| **9** | **ЭЦП на основе симметричной криптографии** | 126 |
| 9.1 | Назначение, структура, особенности построения и использо- |  |
|  | вания ЭЦП | 126 |
| 9.2 | ЭЦП на основе алгоритмов симметричного шифрования | 128 |
| **10** | **ЭЦП на основе асимметричной криптографии** | 130 |
| 10.1 | Особенности ЭЦП на основе алгоритмов асимметричной |  |
|  | криптографии | 130 |
| 10.2 | Основные понятия из области хешировании сообщений | 130 |
| 10.3 | Однонаправленные хеш-функции и алгоритмы хеширования | 132 |
| 10.3.1 | *Алгоритм хеширования MD4* | 132 |
| 10.3.2 | *Особенности алгоритма MD5* | 136 |
| 10.3.3 | *Алгоритмы хеширования семейства SHA* | 137 |
|  |  | 183 |

|  |  |  |
| --- | --- | --- |
| **11** | **ЭЦП на основе хеша сообщения** | 140 |
| 11.1 | ЭЦП при использовании RSA | 140 |
| 11.2 | Схемы ЭЦП на основе проблемы дискретных логарифмов | 141 |
| 11.2.1 | *Алгоритм цифровой подписи DSA* | 142 |
| 11.2.2 | *Схема подписи Эль-Гамаля* | 144 |
| 11.2.3 | *ЭЦП на основе схемы Шнорра* | 145 |
| **12** | **ЭЦП на основе эллиптических кривых** | 147 |
| **13** | **Атаки на ЭЦП и противодействие атакам** | 152 |
| 13.1 | Характеристика основных типов атак | 152 |
| 13.2. | Сертификаты и сертификационные центры | 153 |
|  | **ГЛАВА V МЕТОДЫ ОБФУСКАЦИИ В ЗАЩИТЕ** |  |
|  | **ИНФОРМАЦИИ** |  |
| **14** | **Основные методы защиты программного кода** | 161 |
| 14.1 | Общая сравнительная характеристика методов защиты ПО | 161 |
| 14.2 | Характеристика [современных методов обфускации кода](https://docs.google.com/document/d/1rFFwjOv0QF_ztI5mDmmb4EJcoEYV1WYRGJ-B1s9Sd68/edit#heading=h.pe14fimlal31) | 163 |
| 14.2.1 | *Уровни обфускации* | 164 |
| 14.2.2 | *Особенности применения известных методов обфускации ко-* |  |
|  | *да* | 166 |
| 14.2.2.1 | *Запутывание потока выполнения* | 166 |
| 14.2.2.2 | *Непрозрачные предикаты* | 167 |
| 14.2.2.3 | *Субституция* | 169 |
| 14.2.2.4 | *Кодирование строковых значений* | 169 |
| 14.2.3 | *Использование других форматов хранения* | 172 |
| 14.3 | Авторские методы обфускации | 173 |
| 14.3.1 | *Использование интерпретируемых языков программирования* | 173 |
| 14.3.2 | *Метод шифрования данных* | 177 |
| 14.3.3 | *Метод на основе преобразования кода в нечитабельные инст-* |  |
|  | *рукции* | 179 |
|  | **Оглавление** | 182 |
|  | **Список использованных источников** | 185 |

Приложение

184

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. C.E. Shannon, The Communication Theory of Secrecy Systems//Bell Sys. Tech. J. – V.28. – №4. – 1949. – P.656-715.
  2. Шнаейер, Б. Секреты и ложь. Безопасность данных в цифровом ми-ре / Б. Шнаейер. – СПб.: Питер, 2003. – 370 с.
  3. Шнаейер, Б. Прикладная криптография. Протоколы, алгоритмы, ис-ходные тексты на языке СИ/ Б. Шнаейер. – М.: Издательство Ртиумф, 2003. – 816 с.
  4. Онацкий, А**.** В**.** Асимметричные методы шифрования. – Модуль 2 Криптографические методы защиты информации в телекоммуникационных системах и сетях/А.В. Онацкий, Л.Г. Йона: Учеб. пособие / Под ред. Н**.** В**.** Захарченко – Одесса: ОНАС им. А. С. Попова, 2010. – 148 с.
  5. Криптография, стеганография и охрана авторскогоправа: моногра-фия/ 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 информацион-ных систем/ П.П. Урбанович, Д.В. Шиман. – Минск:БГТУ
  9. Урбанович, П.П. Информационная безопасность и надежность сис-тем/ П.П. Урбанович, Д.М. Романенко, Е.В. Кабак. – Минск:БГТУ
  10. Хогланд, Г. Взлом программного обеспечения. Анализ и использование кода / Г. Хогланд, Г. Мак-Гроу. – М.: Издательский дом «Вильяме», 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\ Пер. с англ.

– 2-е изд., испр. – М.: Издательско-торговый дом «Русская Редакция», 2005. – 704 с.

* 1. Urbanovich, N. The use of steganographic techniques for protection of intellectual property rights / N. Urbanovich, V. Plaskovitsky // New Electrical and Electronic Technologies and their Industrial Implementation. — 2011. — P. 147– 148.
  2. 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, 2011. — P. 189–193.

185

* 1. Пласковицкий, В.А. Защита программного обеспечения от несанк-ционированного использования и модификации методами обфускации/ В.А. Пласковицкий, П.П. Урбанович// Труды БГТУ. Сер. Физико-мат. науки и ин-форматика.- Минск: БГТУ. - 2011.- С.173-176.
  2. Пласковицкий, В.А. Шифрование кодов программ на основе клю-ча, задаваемого рекуррентными математическими соотношениями/ В.А. Пла-сковицкий, П.П. Урбанович// Труды БГТУ. Сер. Физико-мат. науки и информатика.- Минск: БГТУ. – С.146-148.
  3. Urbanovich, P. Text steganography application for protection and transfer of the information/ P. Urbanovich, K. Chourikau, A. Rimorev, N. Urbanovich. – Przeglad elektrotechniczny. — 2012. — № 8. — P. 342–344.
  4. Шутько, Н. П. Особенности и формальное описание процесса осажде-ния секретной информации в текстовые документы на основе стеганографии / Н.
* Шутько // Труды БГТУ. – 2014. – № 6: Физ.-мат. науки и информатика. — С.121-124.
  1. Пласковицкий, В. А. Использование абстракных синтаксических деревьев для обфускации кода / В. А. Пласковицкий, П. П. Урбанович // Тру-ды БГТУ. – 2014. – № 6: Физ. мат. науки и информатика. — С.142-146.
  2. Шутько, Н. П. Математическая модель системы текстовой стегано-графии 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. Интернет-портал [Электронный ресурс]/ Intersections Inc. – Режим доступа:

[http://phx.corporate-ir.net.](http://phx.corporate-ir.net/phoenix.zhtml?c=175233&p=irol-newsArticle&ID=1526456&highlight=) – Дата доступа: 13.02.2015.

1. Интернет-портал [Электронный ресурс]/ Javelin Strategy & Research. – Режим доступа: [https://www.javelinstrategy.com/Brochure-209.](https://www.javelinstrategy.com/Brochure-209) – Дата доступа: 16.09.2015.
2. Интернет-портал [Электронный ресурс]/ Режим доступа: http://bargu.by/658-oxrana-avtorskix-prav-v-seti-internet.html. – Дата доступа: 21.11.2015.
3. Интернет-портал [Электронный ресурс]/ – Режим доступа: <http://csis.org/.../2014-mcafee-report-global-cost->... – Дата доступа: 13.11.2015.
4. Интернет-портал [Электронный ресурс]/ Creation of a global culture of cybersecurity and taking stock of national efforts to protect critical information infrastructures. – Режим доступа:

[http://phx.corporate-ir.net.](http://phx.corporate-ir.net/phoenix.zhtml?c=175233&p=irol-newsArticle&ID=1526456&highlight=) <http://daccess-dds-ny.un.org/doc/UNDOC/GEN/N09/474/49/PDF/> N0947449.pdf? OpenElement. – Дата доступа: 13.02.2015.

1. Интернет-портал [Электронный ресурс]/ The International Obfuscated C Code Contest. – Режим доступа: [http://www.ioccc.org/.](http://www.ioccc.org/) – Дата доступа: 07.05.2014.
2. Интернет-портал [Электронный ресурс] / Inquartos Obfuscator. – Режим доступа: [http://netobf.com/obf\_click\_once\_about.](http://netobf.com/obf_click_once_about) – Дата доступа: 11.05.2014.
3. Интернет-портал [Электронный ресурс] / Крипто-про. – Режим доступа: [http://www.cryptopro.ru.](http://www.cryptopro.ru/) – Дата доступа: 10.05.2013.
4. Интернет-портал [Электронный ресурс] / Dean Packer. – Режим доступа: [http://dean.edwards.name/download/#packer.](http://dean.edwards.name/download/#packer) – Дата доступа: 15.05.2014.

186

1. Интернет-портал [Электронный ресурс] / Официальный сайт Эдварда Тафти. – Режим доступа: [http://www.edwardtufte.com/tufte/.](http://www.edwardtufte.com/tufte/) – Дата доступа: 15.05.2014.
2. Интернет-портал [Электронный ресурс] – Режим доступа: <http://citforum.ru/security/> articles/obfus/. – Дата доступа: 25.12.2015.
3. Интернет-портал [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/> Эллиптическая кривая. – Дата доступа: 14.06.2015.
4. Интернет-портал [Электронный ресурс] – Режим доступа: [https://habrahabr.ru/post/](https://habrahabr.ru/post/%20188958/) [188958/.](https://habrahabr.ru/post/%20188958/) – Дата доступа: 14.06.2015.
5. Интернет-портал [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ Потоковый шифр. – Дата доступа: 19.07.2015.
6. Интернет-портал [Электронный ресурс] – Режим доступа: [http://crypto-r.narod.ru/glava2/glava2\_3.html.](http://crypto-r.narod.ru/glava2/glava2_3.html) – Дата доступа: 19.01.2016.

187

Учебное издание

Урбанович Павел Павлович

ЗАЩИТА ИНФОРМАЦИИ МЕТОДАМИ КРИПТОГРАФИИ,

СТЕГАНОГРАФИИ И ОБФУСКАЦИИ

Учебно-методическое издание

Редактор

Компьютерная верстка

Корректор

Подписано в печать . Формат 60х841/16

Бумага офсетная. Гарнитура Таймс. Печать офсетная.

Усл.печ.л. . Уч.-изд. л.

Тираж 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

188