БАКАЛАВРИАТ и МАГИСТРАТУРА

Е.К. Баранова
А.В. Бабаш

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано
Экспертным советом УМО в системе ВО и СПО
в качестве **учебного пособия** для направления
бакалавриата и магистратуры
«Информационная безопасность»

Второе издание, переработанное и дополненное

BOOK.ru
ЭЛЕКТРОННО-БИБЛИОТЕЧНАЯ СИСТЕМА
КНОРУС • МОСКВА • 2022

+ еПриложение
Дополнительные материалы
BOOK.ru

УДК 681.3(075.8)
ББК 32.81я73
Б24

Рецензенты:
Н.В. Мельников, д-р техн. наук, проф.,
П.Б. Хорев, канд. техн. наук, доц.

Авторы:
Е.К. Баранова, Национальный исследовательский университет «Высшая школа экономики»; Финансовый университет при Правительстве РФ,
А.В. Бабаш, Национальный исследовательский университет «Высшая школа экономики»; РЭУ имени Г.В. Плеханова

Баранова, Елена Константиновна.

Б24 Криптографические методы защиты информации. Лабораторный практикум + еПриложение: дополнительные материалы : учебное пособие / Е.К. Баранова, А.В. Бабаш. — 2-е изд., перераб. и доп. — Москва : КНОРУС, 2022. — 206 с. — (Бакалавриат и магистратура).

ISBN 978-5-406-08831-9

Посвящено рассмотрению практических вопросов защиты информации методами криптографии и стеганографии. В пособие включены описания лабораторных работ с комплектом исполняемых программных модулей. Все практические работы предваряет теоретический раздел и заключает перечень контрольных вопросов.

Соответствует ФГОС ВО последнего поколения.

Для студентов бакалавриата и магистратуры, обучающихся по направлению «Информационная безопасность».

УДК 681.3(075.8)
ББК 32.81я73

+ еПриложение
Дополнительные материалы
BOOK.ru

Дополнительные материалы доступны
на персональной странице издания
в электронно-библиотечной системе BOOK.ru.

Баранова Елена Константиновна
Бабаш Александр Владимирович

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ
Лабораторный практикум

Изд. № 505942. Подписано в печать 18.05.2021. Формат 60×90/16.
Гарнитура «Newton». Печать офсетная.
Усл. печ. л. 11,0. Уч.-изд. л. 9,3. Тираж 500 экз.

ООО «Издательство «КноРус».
117218, г. Москва, ул. Кедрова, д. 14, корп. 2.
Тел.: +7 (495) 741-46-28.
E-mail: welcome@knorus.ru www.knorus.ru

Отпечатано в АО «Т8 Издательские Технологии».
109316, г. Москва, Волгоградский проспект, д. 42, корп. 5.
Тел.: +7 (495) 221-89-80.

ISBN 978-5-406-08831-9

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Информационная безопасность является одной из главных проблем, с которой сталкивается современное общество. Причиной обострения этой проблемы является широкомасштабное использование автоматизированных средств накопления, хранения, обработки и передачи информации.

Решение проблемы информационной безопасности связано с гарантированным обеспечением трех ее главных составляющих: *доступности*, *целостности* и *конфиденциальности информации*.

Поскольку информация перестала быть просто необходимым для производства вспомогательным ресурсом или побочным проявлением всякого рода деятельности, а приобрела ощутимый стоимостной вес, который четко определяется реальной прибылью, получаемой при ее использовании, или размерами ущерба, с разной степенью вероятности наносимого владельцу информации в случае ее искажения или утраты. Обеспечение целостности, доступности и конфиденциальности информации, циркулирующей в информационно-вычислительных системах и сетях, приобрели в настоящее время исключительное значение.

Предлагаемый практикум включает в себя 15 методических описаний лабораторных работ с комплектом исполняемых модулей. Весь практикум разделен на четыре части с той целью, чтобы при проведении занятий преподаватель, в зависимости от количества часов выделенных на проведение занятий и своего видения курса, мог выбирать из предложенных разделов те или иные работы.

В первую часть включены практические задания, связанные с методами шифрования, использующими классические симметричные алгоритмы; исследованием различных методов защиты текстовой информации и их стойкости на основе подбора ключей; изучением устройства и принципа действия шифровальной машины Энигма, с использованием программного эмулятора; изучением современного стандарта симметричного шифрования *AES Rijndael*.

При выполнении практических заданий из первой части предполагается рассмотрение следующих вопросов.

Изучение классических криптографических алгоритмов моноалфавитной подстановки, многоалфавитной подстановки и перестановки для защиты текстовой информации. Использование гистограмм, отображающих частоту встречаемости символов в тексте для криптоанализа классических шифров (лабораторная работа № 1).

Изучение методов шифрования (расшифрования) перестановкой символов, подстановкой, гаммированием, использованием таблицы

Виженера. Исследование и сравнение стойкости различных методов, на основе атак путем перебора всех возможных ключей (лабораторная работа № 2).

Изучение принципов шифрования (расшифрования) информации, используемых в шифровальной машине Энигма. Ознакомление с общими принципами действия шифровальной машины Энигма на примере эмулятора *Enigma* (лабораторная работа № 3).

Ознакомление с принципами шифрования, используемыми в алгоритме симметричного шифрования *AES Rijndael* (лабораторная работа № 4).

Вторая часть включает в себя практические задания для изучения процессов генерации простых чисел для систем асимметричного шифрования; процессов постановки и верификации электронной цифровой подписи; исследования шифра скользящей перестановки; изучения пакета *PGP* — программного обеспечения для защиты конфиденциальной информации.

При выполнении практических заданий из второй части предполагается рассмотрение следующих вопросов.

Изучение методов генерации простых чисел, используемых в системах шифрования с открытым ключом (лабораторная работа № 5).

Знакомство с основными положениями Федеральной целевой программы «Электронная Россия». Ознакомление с принципами защищенного электронного документооборота в телекоммуникационных сетях и алгоритмами постановки электронной цифровой подписи (лабораторная работа № 6).

Исследование шифра скользящей перестановки с использованием программной реализации *XY-Mover* (лабораторная работа № 7).

Ознакомление с общими принципами построения и использования программных средств защиты информации, в частности с программой *PGP* (*Pretty Good Privacy*).

Освоение средств программной системы *PGP*, предназначенных:

- для шифрования конфиденциальных ресурсов и разграничения доступа к ним;
- обеспечения целостности информационных ресурсов с помощью механизма электронной цифровой подписи;
- надежного уничтожения остаточной конфиденциальной информации;
- скрытия присутствия в компьютерной системе конфиденциальной информации с помощью виртуального диска (лабораторная работа № 8).

При выполнении практических заданий из третьего раздела рассматриваются следующие вопросы.

Изучение принципа шифрования информации с помощью биграммного шифра Плейфера (лабораторная работа № 9).

Дешифрование шифра простой перестановки при помощи метода биграмм (лабораторная работа № 10).

Знакомство с принципом работы сети Фейстеля, как базовым преобразованием симметричных блочных криптосистем (лабораторная работа № 11).

Изучение принципа работы генератора псевдослучайных последовательностей, основанного на регистре сдвига с линейной обратной связью (лабораторная работа № 12).

В четвертую часть практикума включены три лабораторные работы, предполагающие изучение защиты информационных ресурсов методами стеганографии. Стеганографическая система или стегосистема — совокупность средств и методов, которые используются для формирования скрытого канала передачи информации. Таким образом в отличие от криптографических методов методы стеганографии предполагают скрытие самого факта передачи информации.

Авторы не вынесли в заголовок практикума изучение стеганографических методов защиты, но, поскольку методы стеганографии становятся чрезвычайно популярными в настоящее время и во многом, несмотря на нормативно-правовые трудности их применения, становятся альтернативой криптографическим методам защиты, авторы посчитали необходимым ознакомить читателей с основами использования стеганографии для защиты информационных ресурсов. Вопросам стеганографической защиты посвящены следующие лабораторные работы.

Ознакомление с методами защиты исполняемых программных файлов. Изучение возможностей стеганографической защиты *.exe* файлов путем встраивания цифровых водяных знаков (ЦВЗ) в пустое место, в конце секции файла (лабораторная работа № 13).

Защита электронных документов с использованием цифровых водяных знаков на примерах встраивания ЦВЗ в бинарные изображения (лабораторная работа № 14).

Знакомство с программным обеспечением для стеганографической защиты информации на примере программного пакета *Invisible Secrets*–4 (лабораторная работа № 15).

В книгу вошли материалы лабораторных работ, которые проводились авторами на протяжении последних лет для студентов Националь-

ного исследовательского университета «Высшая школа экономики» (НИУ ВШЭ), Российского государственного социального университета (РГСУ) и Московского государственного университета экономики, статистики и информатики (МЭСИ). Большая часть демонстрационных программ для исследования процессов защиты информации написаны студентами этих вузов под руководством авторов в рамках курсовых и дипломных работ. Авторы выражают благодарность своим студентам: Суханову Дмитрию, Беловой Ирине, Тащилину Евгению, Александрову Артему, Чудакову Сергею, Лазареву Вячеславу, Ларионову Игорю и другим создателям демонстрационных программ. А также тестировщикам программного обеспечения: Корниенко Анастасии, Галинову Андрею, Мочалину Анатолию и Фроловой Анне.

Лабораторный практикум рассчитан, в первую очередь, на студентов бакалавриата высших учебных заведений и магистрантов, обучающихся по различным специальностям, использующих федеральный компонент по основам информационной безопасности и защите информации.

ЧАСТЬ 1

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Проблемы защиты информации. В настоящее время во всем мире резко повысилось внимание к проблеме информационной безопасности. Это обусловлено процессами стремительного расширения потоков информации, пронизывающих все сферы жизни общества.

Информация давно перестала быть просто необходимым для производства вспомогательным ресурсом или побочным проявлением всякого рода деятельности. Она приобрела ощутимую стоимость, которая определяется реальной прибылью, получаемой при ее использовании, или размерами ущерба с разной степенью вероятности наносимого владельцу информации. Однако создание индустрии переработки информации порождает целый ряд сложных проблем. Одной из таких проблем является надежное обеспечение сохранности и установленного статуса информации, циркулирующей и обрабатываемой в информационно-вычислительных системах и сетях.

Появление глобальных компьютерных сетей сделало простым доступ к информации как отдельным пользователям, так и большим организациям. Однако легкость и высокая скорость доступа к данным с помощью таких компьютерных сетей, как Internet, также сделали значительными следующие угрозы безопасности данных при отсутствии мер их защиты:

- неавторизованный доступ к информации;
- неавторизованное изменение информации;
- неавторизованный доступ к сетям и сервисам;
- другие сетевые атаки, например повтор перехваченных ранее транзакций и атаки типа «отказ в обслуживании».

При обработке любой значимой информации при помощи отдельного компьютера, а тем более в сети, возникает вопрос о ее защите от несанкционированного доступа и использования. Наиболее распространенный в компьютерных системах способ защиты — использование паролей — более пригоден для защиты доступа к вычислительным ресурсам, нежели для защиты информации. Пароль — своеобразный экран, отгораживающий законных пользователей системы от посто-

ронних, пройдя сквозь который санкционированный пользователь получает доступ практически ко всей информации.

В настоящее время исключительно важное значение в разных областях приобрели вопросы, связанные с сохранением и передачей конфиденциальной информации. Возникающие при этом задачи решает *криптография* — наука о методах преобразования информации в целях ее защиты от незаконных пользователей.

Ретроспективный взгляд на историю развития криптографии как специфическую область человеческой деятельности позволяет выделить три основных периода. Первый, наиболее продолжительный, — период «ручной криптографии». Его начало теряется в глубокой древности, а закончился он в 1930-е гг. Криптография прошла путь от магического искусства до вполне прозаической прикладной специальности чиновников дипломатических и военных ведомств.

Второй период — создание и широкое внедрение в практику сначала механических, затем электромеханических и электронных устройств шифрования, организация целых сетей засекреченной связи. Его началом можно считать разработку Гилбертом Вернамом (G. Vernam) в 1917 г. схемы телеграфной шифровальной машины, использующей одноразовую гамму (рис.1.1).

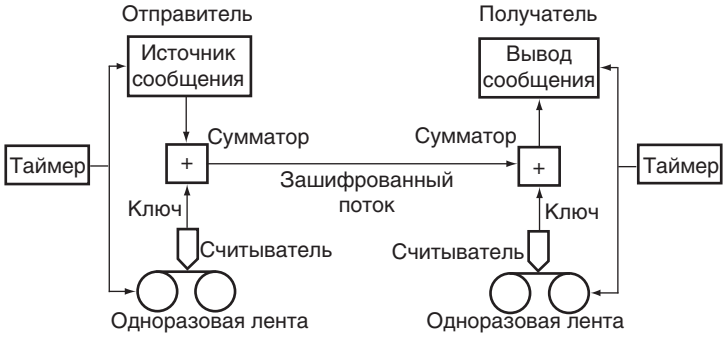


Рис. 1.1. Схема шифрования методом Вернама

К середине 1970-х гг. с развитием разветвленных коммерческих сетей связи, электронной почты и глобальных информационных систем на первый план вышли новые проблемы — снабжения ключами и подтверждения подлинности.

В 1976 г. американские ученые Уитфилд Диффи (W. Diffie) и Мартин Хеллман (M. Hellman) предложили два новых принципа организации засекреченной связи без предварительного снабжения

Рис. 1.2. Джон Валлис

абонентов секретными ключами шифрования — принцип так называемого *открытого шифрования* и принцип *открытого распределения ключей*. Этот момент можно считать началом нового периода в развитии криптографии. В настоящее время это направление современной криптографии очень интенсивно развивается.

Из истории криптографии. Понятие «безопасность» охватывает широкий круг интересов как отдельных лиц, так и целых государств. Во все исторические времена существенное внимание уделялось проблеме информационной безопасности, обеспечению защиты конфиденциальной информации от ознакомления, кражи, модификации, подмены. Решением этих вопросов занимается криптография.

Термин «криптография» (тайнопись) ввел английский математик Джон Валлис (John Wallis) (1616—1703) (рис. 1.2). Потребность шифровать и передавать шифрованные сообщения возникла очень давно.

Сцитала. Еще в V—VI вв. до н.э. греки использовали специальное шифрующее устройство. По описанию Плутарха, это устройство состояло из двух цилиндрических стержней одинаковой длины и толщины, которые называли *сциталами* (рис. 1.3). При необходимости передачи сообщения длинную ленту папируса наматывали на сциталу, не оставляя на ней никакого промежутка, и писали на нем необходимую информацию. Затем папирус снимали и без стержня отправляли адресату. Поскольку буквы оказывались разбросанными в беспорядке, то прочитать сообщение мог только тот, кто имел свою сциталу такой же длины и толщины, чтобы намотать на нее папирус.

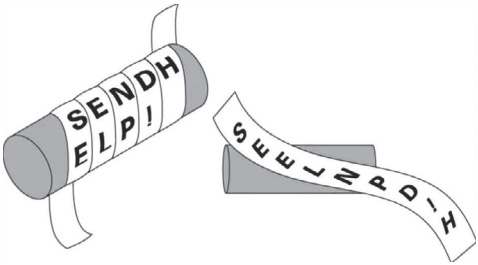


Рис. 1.3. Сцитала

Квадрат Полибия[1]**.** В Древней Греции (II в. до н.э.) был известен шифр, называемый *квадратом Полибия*. Это устройство представляло собой квадрат 5×5, столбцы и строки которого нумеровали цифрами от 1 до 5. В каждую клетку записывалась одна буква (в греческом варианте одна клетка оказывалась пустой, а в латинском — в одну клетку помещали две буквы: I, J). В результате каждой букве отвечала пара чисел по номеру строки и столбца.

а *б*

Пример квадрата Полибия приведен на рис. 1.4.

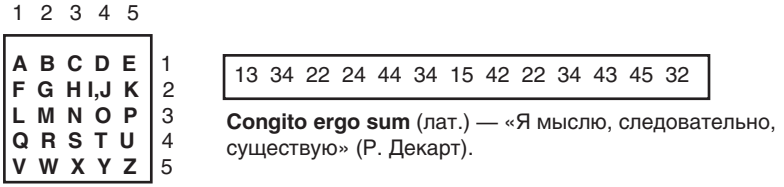


Рис. 1.4. Квадрат Полибия (*а*) и пример шифрования (*б*)

Код Цезаря. В I веке н.э. Ю. Цезарь во время войны с галлами, пере-

а *б*

писываясь со своими друзьями в Риме, заменял в сообщении первую букву латинского алфавита (A) на четвертую (D), вторую (B) — на пятую (E), наконец, последнюю — на третью.

Пример кода Цезаря изображен на рис. 1.5.

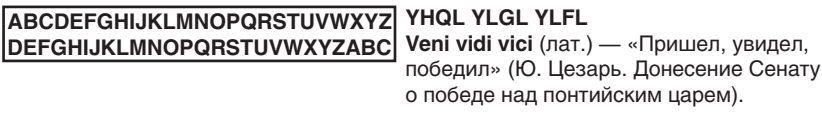


Рис. 1.5. Код Цезаря (*а*) и пример шифрования (*б*)

Шифр Цезаря относится к так называемому классу *моноалфавитных подстановок* и имеет множество модификаций.

Решетка Кардано. Широко известны шифры, относящиеся к классу *перестановки,* в частности «*решетка Кардано*»[2]. Это прямоугольная карточка с отверстиями, чаще всего квадратная, которая

[1] Полибий (200—120 гг. до н.э.) — древнегреческий историк.

[2] Кардано Джероламо (1501—1576) — итальянский математик, философ и врач.

при наложении на лист бумаги оставляет открытыми лишь некоторые его части. Число строк и столбцов на карточке четное. Карточка сделана так, что при последовательном ее поворачивании каждая клетка лежащего под ней листа окажется занятой. Карточку поворачивают сначала вдоль вертикальной оси симметрии на 180°, а затем вдоль горизонтальной оси также на 180° (рис. 1.6). И вновь повторяют ту же процедуру.

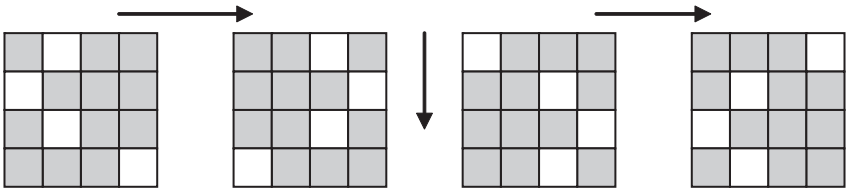


Рис. 1.6. Решетка Кардано

Диск Альберти. Итальянец Альберти (XVI в.) впервые выдвинул идею двойного шифрования — текст, полученный в результате первого шифрования, подвергался повторному шифрованию. В трактате Альберти был приведен его собственный шифр, который он назвал «шифром, достойным королей». Он утверждал, что этот шифр недешифруем. Реализация шифра осуществлялась с помощью шифровального диска, положившего начало целой серии *многоалфавитных подстановок*. Устройство представляло собой пару дисков — внешний, неподвижный (на нем были нанесены буквы в естественном порядке и цифры от 1 до 4) и внутренний — подвижный (на нем буквы были переставлены) (рис. 1.7). Процесс шифрования заключался в нахождении буквы открытого текста на внешнем диске и замену ее на соответствующую (стоящую под ней) букву шифрованного текста. После шифрования нескольких слов внутренний диск сдвигался на один шаг. Ключом данного шифра являлся порядок расположения букв на внутреннем диске и его начальное положение относительно внешнего диска.

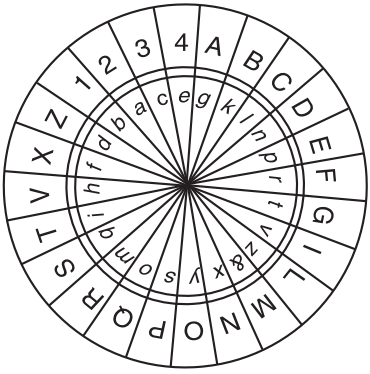


Рис. 1.7. Диск Альберти

Таблица Виженера[1]. Неудобство рассмотренных выше шрифтов моноалфавитных подстановок очевидно, так как в случае использования стандартного алфавита таблица частот встречаемости букв алфавита позволяет определить один или несколько символов, а этого иногда достаточно для вскрытия шифра («Пляшущие человечки» Конан Дойля или «Золотой жук» Эдгара По). Поэтому, для того чтобы затруднить дешифрование, использовали различные приемы, например *таблицу Виженера*, представляющую собой квадратную таблицу с числом строк и столбцов, равным количеству букв алфавита (рис 1.8). Чтобы зашифровать какое-либо сообщение, выбирают слово-лозунг (например, «монастырь») и надписывают его над сообщением с необходимым повторением.

Чтобы получить шифрованный текст, находят очередной знак лозунга, начиная с первого, в вертикальном алфавите, а ему — соответствующий знак сообщения в горизонтальном алфавите. На пересечении выделенных столбца и строки находят первую букву шифра. Очевидно, что ключом к такому шифру является используемый лозунг.

Одноразовый шифровальный блокнот. Примером нераскрываемого шифра может служить так называемый *одноразовый шифровальный блокнот* — шифр, в основе которого лежит та же идея, что и в шифре Цезаря. Назовем *расширенным алфавитом* множество букв алфавита и знаков препинания { . , : ; ! ? () — “ <пробел>}, число символов расширенного кириллического алфавита в данном варианте будет равно 44. Занумеруем символы расширенного алфавита числами от 0 до 43. Тогда любой передаваемый текст можно рассматривать как последовательность $\{a_n\}$ чисел множества $A = \{0, 1, ..., 43\}$.

Предположим, что имеем случайную последовательность $\{c_n\}$ из чисел множества A той же длины, что и передаваемый текст — *ключ*. Складывая по модулю 44 число из передаваемого текста a_n с соответствующим числом из множества ключа c_n:

$$a_n + c_n \equiv b_n \pmod{44},\ 0 \le b_n \le 43,$$

получим последовательность $\{b_n\}$ знаков шифрованного текста. Чтобы получить передаваемый текст, можно воспользоваться тем же ключом:

$$a_n \equiv b_n - c_n \pmod{44},\ 0 \le a_n \le 43.$$

[1] Блез де Виженер (1523—1596) — французский посол в Риме, который написал большой труд о шифрах. Квадратный шифр Виженера на протяжении почти 400 лет не был дешифрован и считался недешифруемым шифром.

А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я
Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А
В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б
Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В
Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я Ф Б В Г
Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д
Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е
З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж
И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З
Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И
К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й
Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К
М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л
Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М
О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н
П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О
Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П
С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р
Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С
У Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т
Ф Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У
Х Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф
Ц Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х
Ч Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц
Ш Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч
Щ Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш
Ь Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ
Ы Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь
Э Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Щ Ш Ь Ы
Ю Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э
Я А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Э Ю

а

м о н а с т ы р ь м о н а с т ы р ь м о н

р а с к и н у л о с ь м о р е ш и р о к о

э о я к щ а п ы й ю й щ о в ч ф ш л ь ш ы

б

Рис. 1.8. Таблица Виженера (*а*) и пример шифрования (*б*)

У двух абонентов, находящихся в секретной переписке, имеются два одинаковых блокнота. В каждом из них на нескольких листах напечатана случайная последовательность чисел множества *A*. Отправитель свой текст шифрует указанным выше способом при помощи первой страницы блокнота. Зашифровав сообщение, он уничтожает использованную страницу и отправляет текст сообщения второму абоненту. Получатель шифрованного текста расшифровывает его и также уничтожает использованный лист блокнота. Очевидно, что одноразовый шифр не раскрываем в принципе, так как символ в тексте может быть заменен любым другим символом и этот выбор совершенно случаен.

Методы шифрования. ***Одноалфавитный метод***[1]**.** Данный метод, пожалуй, самый древний из всех известных методов. В его основе лежит простой способ шифрования: отправитель и получатель зашифрованного документа заранее договариваются об определенном смещении букв относительно их обычного местоположения в алфавите. Например, для кириллицы если смещение равно 1, то буква «А» соответствует букве «Б», «Б» — «В» и т.д. Когда алфавит заканчивается, берут буквы из начала списка. И выходит, например, следующее: из слова КОДИРОВАНИЕ получается ЛПЕЙСПГБОЙЖ.

Частным случаем этого является ранее рассмотренный шифр Цезаря. Очевидно, что произвольный шифр из класса одноалфавитных методов не является шифром Цезаря (если мощность алфавита текста равна n, то число шифров Цезаря равно n, а число всех одноалфавитных шифров — $n!$). Однако и для таких методов легко предложить способы дешифрования, основанные на статистических свойствах шифрованных текстов, поскольку открытый и закрытый тексты имеют одинаковые статистические характеристики.

Шифрование методом перестановки символов. Суть этого метода заключается в том, что символы текста переставляются по определенным правилам, при этом используются только символы исходного (незашифрованного) текста. Перестановки в классической криптографии обычно получают в результате записи исходного текста и чтения шифрованного текста по разным путям геометрической фигуры. Простейшим примером перестановки является запись исходного текста по строкам некоторой матрицы и чтение его по столбцам этой матрицы.

Последовательность заполнения строк и чтения столбцов может быть любой и задается ключом. Таким образом, для матрицы размером

[1] В лабораторной работе № 1 рассматриваются два варианта одноалфавитного метода: с фиксированным смещением и с произвольным (задаваемым) смещением.

8×8 (длина блока 64 символа) возможно $1{,}6\times10^9$ ключей, что позволяет на современных компьютерах путем перебора дешифровать текст. Однако для матрицы размером 16×16 (длина блока 256 символов) существует $1{,}4\times10^{26}$ ключей и перебор их с помощью современных вычислительных средств весьма затруднителен.

Примером применения метода перестановки символов является восьмиэлементная таблица, обладающая совокупностью маршрутов, которые называются «маршруты Гамильтона». Последовательность заполнения таблицы каждый раз соответствует нумерации ее элементов. Если длина шифруемого текста не кратна числу элементов, то при последнем заполнении в свободные элементы заносится произвольный символ. Выборка из таблицы для каждого заполнения может выполняться по своему маршруту, при этом маршруты могут использоваться как последовательно, так и в порядке, задаваемом ключом.

Для методов перестановки характерны простота алгоритма, возможность программной реализации и низкий уровень защиты, так как при большой длине исходного текста в его зашифрованном варианте проявляются статистические закономерности ключа, что и позволяет его быстро раскрыть. Другой недостаток этих методов — легкое раскрытие, если удается направить в систему для шифрования несколько специально подобранных сообщений. Так, если длина блока в исходном тексте равна K символам, то для раскрытия ключа достаточно пропустить через шифровальную систему $K - 1$ блоков исходного текста, в которых все символы, кроме одного, одинаковы.

***Шифрование инверсными символами* (*по дополнению до 255*).** Данный метод шифрования является частным случаем одноалфавитной замены в алфавите мощности 256. Суть метода заключается в замене символа ASCII-кодировки с номером i на символ с номером $255 - i$. Аналогично проводится и операция расшифрования.

Многоалфавитные методы шифрования[1]. Многоалфавитное шифрование (многоалфавитная замена) заключается в том, что для последовательных символов шифруемого текста используются одноалфавитные методы с различными ключами.

Например, первый символ заменяется по методу Цезаря со смещением 14, второй — со смещением 10 и т.д. до конца заданного ключа. Затем процедура продолжается периодически. Более общей является ситуа-

[1] В лабораторной работе № 1 рассматриваются три варианта многоалфавитного метода: с фиксированным ключом, с ключом фиксированной длины и с ключом произвольной длины.

ция, когда используется не шифр Цезаря, а последовательность произвольных подстановок, соответствующих одноалфавитным методам.

Более наглядным примером подобного шифрования является метод гаммирования. Этот способ преобразования заключается в том, что символы закрываемого текста последовательно складываются с символами некоторой специальной последовательности, именуемой гаммой. Такое преобразование иногда называют наложением гаммы на открытый текст.

Собственно процедура наложения может осуществляться одним из двух способов:

1) символы закрываемого текста и гаммы заменяются цифровыми эквивалентами, а затем складываются по модулю K:

$$T_{ш} = (T_{о} \oplus T_{г}) \bmod K,$$

где $T_{ш}$ — шифротекст; $T_{о}$ — открытый текст; $T_{г}$ — гамма; K — количество символов алфавита;

2) символы текста и гаммы представляются в двоичных кодах, а затем каждая пара двоичных разрядов складывается по mod 2.

Стойкость шифрования методом гаммирования определяется главным образом качеством гаммы, которое характеризуется длиною периода и случайностью распределения по периоду.

Длинна периода гаммы — минимальное количество символов, после которого последовательность начинает повторяться. Случайность распределения символов по периоду означает отсутствие закономерностей между появлением различных символов в пределах периода.

Основные требования, предъявляемые к методам шифрования информации:

■ сложность и трудоемкость процедур шифрования и расшифрования должны определяться в зависимости от степени секретности защищаемых данных;

■ надежность закрытия должна быть такой, чтобы секретность не нарушалась даже в том случае, когда злоумышленнику известен способ закрытия;

■ способ закрытия и набор используемых служебных данных (ключевых установок) не должны быть слишком сложными. Затраты на защитные преобразования должны быть приемлемые при заданном уровне сохранности информации;

■ выполнение процедур прямого и обратного преобразования должно быть формальным и как можно проще;

■ процедуры прямого и обратного преобразования не должны зависеть от длины сообщения;

■ ошибки, возникающие в процессе преобразования, не должны распространяться по системе и вызывать потерю информации. Из-за появления ошибок передачи зашифрованного сообщения по каналам связи не должна исключаться возможность надежной расшифровки текста на приемном конце;

■ избыточность сообщений, вносимая закрытием, должна быть как можно меньшей;

■ объем ключа не должен затруднять его запоминание и пересылку.

Гистограмма текста[1]**.** Одним из наиболее известных методов криптоанализа является изучение статистических характеристик шифрованных текстов. Графическое отображение совокупности частот встречаемости символов в тексте называют гистограммой этого текста.

Предположим, что мы имеем дело с методом одноалфавитного шифрования. Зная частоту встречаемости букв в алфавите, можно предположить, какая буква была заменена на данную. Например, часто встречаемая буква «О» заменена на редко встречающуюся букву «Щ».

Следует иметь в виду, что вид гистограммы для стандартного распределения зависит от вида исходного текста следующим образом: если исходный текст содержит символы кириллицы и латинского алфавита, то выводится статистическое распределение для кириллицы и латиницы, если только кириллицы (латиницы), то выводится статистическое распределение для кириллицы (латиницы).

[1] Для наглядности в лабораторной работе № 1 используются двойные гистограммы, отображающие частоту встречаемости символов в исходном и зашифрованном текстах.

ЛАБОРАТОРНАЯ РАБОТА № 1

ИСПОЛЬЗОВАНИЕ КЛАССИЧЕСКИХ КРИПТОАЛГОРИТМОВ ПОДСТАНОВКИ И ПЕРЕСТАНОВКИ ДЛЯ ЗАЩИТЫ ТЕКСТОВОЙ ИНФОРМАЦИИ

Цель работы: изучение классических криптографических алгоритмов моноалфавитной подстановки, многоалфавитной подстановки и перестановки для защиты текстовой информации. Использование гистограмм, отображающих частоту встречаемости символов в тексте для криптоанализа классических шифров.

Описание лабораторной работы. Для выполнения лабораторной работы необходимо запустить программу **L_LUX.EXE**. На экране дисплея появляется окно с размещенным в его центре текстовым редактором (для отображения зашифрованных и расшифрованных текстов), в верхней строке окна расположено главное меню, позволяющее пользователю выполнить требуемое действие. Чуть ниже основного меню размещена панель инструментов (для управления быстрыми командными кнопками и другими «горячими» элементами управления), а в самом низу окна, под текстовым редактором, находится строка состояния, в которой указывается подсказка и выводится дополнительная информация. Клавиши панели инструментов для удобства снабжены всплывающими подсказками.

Для того чтобы попасть в основное меню, необходимо нажать клавишу F10. Передвижение по главному меню осуществляется клавишами перемещения курсора. Чтобы вызвать пункт меню, нужно нажать клавишу ENTER, вернуться в главное меню или вовсе выйти из него — ESC.

Рассмотрим более подробно каждый из пунктов главного меню.

Редактор. Данный пункт основного меню содержит подпункты: создать документ, открыть файл, сохранить файл, выход из программы.

Предварительно, сразу после запуска программы, текстовый редактор недоступен, также недоступными являются почти все пункты главного меню, кроме создания документа, открытия файла, выхода из программы, информации о программе, и бóльшая часть клавиш панели управления, за исключением создания документа, открытия файла и выхода из программы.

Создать документ (Ctrl+N) — при выборе данного подпункта становится доступна работа с тестовым редактором (пользователь получает право создать свой текстовый файл, который впоследствии можно будет использовать при работе с программой), также появляется возмож-

ность использовать все недоступные до этого пункты основного меню и клавиши панели управления.

Открыть файл (Ctrl+L) — при выборе этого пункта появляется диалоговое окно, предоставляющее возможность выбора файла для загрузки. При этом содержимое файла будет отображено в окне редактора текстов. Для корректного отображения содержимого текстовые файлы должны быть в кодировке ANSI (Windows-1251).

Аналогично пункту СОЗДАТЬ ДОКУМЕНТ доступным для работы становится текстовый редактор с отображаемым текстом, а также появляется возможность использовать все недоступные до этого пункты основного меню и клавиши панели управления.

Сохранить файл (Ctrl+S) — при выборе этого пункта появляется диалоговое окно, позволяющее сохранить на диске содержимое редактора текстов.

Выход из программы (Ctrl+X) — при выборе этого пункта появляется диалоговое окно, позволяющее выйти из программы.

Гистограмма. Вывод на экран двух гистограмм, отображающих частоту встречаемости символов в тексте.

Внимание! До выполнения шифрования и дешифрования вызывать гистограмму не имеет смысла, так как еще не сформированы тексты, для которых будет просматриваться гистограмма.

Имеется возможность просмотра следующих сочетаний гистограмм:

- исходного и зашифрованного файла;
- зашифрованного и расшифрованного файла;
- стандартного распределения и зашифрованного текста;
- стандартного распределения и расшифрованного текста.

С целью масштабирования в гистограммах используются левая и правая клавиши мыши. Например, после шифрования текста большого объема пользователь хочет посмотреть гистограммы исходного и зашифрованного файла. Поскольку размеры текста достаточно большие, то на экран будут выведены две гистограммы с большим количеством столбцов в каждой (столбец соответствует одному символу текста), однако трудно будет сказать, какой из этих столбцов соответствует тому или иному символу текста и какова частота встречаемости данного символа. Поэтому у пользователя есть возможность увеличить масштаб любой из двух гистограмм для более точного определения требуемого значения частоты встречаемости конкретного символа. Для этого необходимо навести указатель мыши на левую границу того участка гистограммы, который требуется увеличить, затем нажать левую клавишу мыши и, не отпуская ее, растянуть прямоугольник так,

чтобы его нижний правый угол совпал с правой границей увеличиваемого участка гистограммы. После этого следует отпустить левую клавишу мыши и на экране появится увеличенное изображение нужного участка. Нажав и не отпуская правую клавишу мыши, можно перемещать гистограмму в любом направлении с целью изучения всего полученного распределения в увеличенном масштабе.

Для того чтобы от увеличенного масштаба вернуться к исходному виду, нужно навести указатель мыши на гистограмму, затем нажать левую клавишу мыши и, не отпуская ее, снизу вверх растянуть небольшой по размерам прямоугольник, после этого следует отпустить левую клавишу мыши и на экране появится исходное изображение гистограммы.

Шифрование. Выполнение шифрования текстового файла осуществляется одним из семи методов, рассматриваемых в лабораторной работе:

1) одноалфавитный (с фиксированным смещением);
2) одноалфавитный с задаваемым смещением (от 2 до 20);
3) перестановка символов;
4) по дополнению до 255 (инверсный);
5) многоалфавитный (с фиксированным ключом);
6) многоалфавитный с ключом фиксированной длины;
7) многоалфавитный с ключом произвольной длины.

Выбор метода шифрования производится как мышкой, так и клавишами перемещения курсора и клавишей ENTER. Затем появляется окно, в котором в зависимости от метода шифрования требуется указать те или иные параметры и либо подтвердить процесс кодировки, либо отказаться от него. После этого в окне редактора будет выдан зашифрованный текст.

Расшифрование. Аналогично предыдущему пункту выбирается метод расшифрования (должен соответствовать методу, которым был зашифрован файл). Снова появляется окно, в котором в зависимости от метода расшифрования требуется указать те или иные параметры и либо подтвердить процесс расшифрования, либо отказаться от него. После этого в окно редактора будет выдан расшифрованный текст. При правильном расшифровании полученный текст совпадает с исходным.

Дополнительная информация. Программа предусматривает возможность посмотреть дополнительную информацию («Помощь Ctrl+F9»), справочную информацию об используемых методах шифрования («О методах Ctrl+F10»), сведения о программе («О программе Ctrl+F11»).

Пример работы с программой

Рассмотрим одноалфавитное шифрование с фиксированным ключом.

Нажав клавиши Ctrl+L либо выбрав в меню пункт Открыть файл, загрузите в окно редактора исходный текст.

Затем вызовите пункт меню Шифрование, выберите одноалфавитный метод (с фиксированным смещением). В появившемся окне нажмите клавишу Зашифровать. После того как шифрование выполнено, можно в редакторе просмотреть зашифрованный текст.

Перейдите к пункту меню Гистограмма. Выберите тип гистограмм, отображающий гистограммы исходного и шифрованного файлов. Проанализируйте гистограммы. Они должны иметь примерно одинаковый вид.

Чтобы узнать ключ шифра (смещение второго алфавита относительно первого), необходимо по гистограммам найти символы, имеющие одинаковую частоту встречаемости. Например, самый частый символ в первой гистограмме при шифровании должен перейти в самый частый символ во второй гистограмме. Таким образом, найдя два самых часто встречаемых символа в обеих гистограммах, можно по стандартной таблице ASCII-кодов вычислить смещение. Зная смещение и таблицу кодировки символов, текст можно легко расшифровать. Вызвав пункт меню Дешифрование, можно провести те же действия в автоматическом режиме.

Примечание. При шифровании и расшифровании из таблицы кодировки не используются символы с кодами 176—223 и 240—255, т.е. при ручной расшифровке эти символы следует пропускать и считать, что, например, символ «Я» имеет код не 159, а 223, аналогично «п» не 175, а 239.

Иногда в гистограммах под столбцами, показывающими частоту встречаемости символов, изображены не сами символы, а их табличные коды в квадратных скобках.

Ниже приведено описание «горячих» клавиш и их использование при выполнении различных действий:

Shift+F10 — о программе;
Ctrl+X — выход из программы;
Ctrl+N — new — ФАЙЛ/Создать;
Ctrl+L — load — Файл/Открыть;
Ctrl+S — save — Файл/Сохранить.
Шифрование:
Ctrl+F1 — одноалфавитный метод (с фиксированным смещением);
Ctrl+F2 — одноалфавитный с задаваемым смещением (от 2 до 20);

Ctrl+F3 — перестановка символов;
Ctrl+F4 — по дополнению до 255 (инверсный метод);
Ctrl+F5 — многоалфавитный метод с фиксированным ключом;
Ctrl+F6 — многоалфавитный метод с ключом фиксированной длины;
Ctrl+F7 — многоалфавитный метод с ключом произвольной длины.

Расшифрование:

Shift+F1 — одноалфавитный метод (с фиксированным смещением);
Shift+F2 — одноалфавитный с задаваемым смещением (от 2 до 20);
Shift+F3 — перестановка символов;
Shift+F4 — по дополнению до 255 (инверсный метод);
Shift+F5 — многоалфавитный метод с фиксированным ключом;
Shift+F6 — многоалфавитный метод с ключом фиксированной длины;
Shift+F7 — многоалфавитный метод с ключом произвольной длины.

Гистограммы:

Shift+Ctrl+F1 — исходного и шифрованного файла;
Shift+Ctrl+F2 — шифрованного и расшифрованного файла;
Shift+Ctrl+F3 — стандартного распределения и шифрованного текста;
Shift+Ctrl+F4 — стандартного распределения и расшифрованного текста.

Помощь:

Ctrl+F9 — помощь;
Ctrl+F10 — о методах;
Ctrl+F11 — о программе.

Задание

1. Ознакомиться с описанием лабораторной работы и заданием.
2. Для одноалфавитного метода с фиксированным смещением определить установленное в программе смещение. Для этого следует:

- просмотреть предварительно созданный с помощью редактора свой текстовый файл;
- выполнить для этого файла шифрование;
- просмотреть в редакторе зашифрованный файл;
- просмотреть гистограммы исходного и зашифрованного текстов;
- описать гистограммы (в чем похожи, чем отличаются) и определить, с каким смещением было выполнено шифрование;

- расшифровать зашифрованный текст:
  - с помощью программы, после чего проверить в редакторе правильность расшифрования,
  - вручную с помощью гистограмм; описать и объяснить процесс дешифрования.

В отчете для каждого метода шифрования описывается последовательность выполняемых действий, имена всех использованных файлов, полученные гистограммы, указывается найденное смещение, описывается процесс дешифрования.

Преподавателю предоставляется отчет о проделанной работе и все использованные и созданные файлы.

3. Для одноалфавитного метода с задаваемым смещением (шифр Цезаря) следует:

- выполнить шифрование с произвольным смещением для своего исходного текста;
- просмотреть и описать гистограммы исходного и зашифрованного текстов, определить смещение для нескольких символов;
- расшифровать текст с помощью программы;
- дешифровать зашифрованный шифром Цезаря текст с помощью программы методом подбора смещения; указать, с каким смещением был зашифрован файл.

4. Для метода перестановки символов дешифровать зашифрованный файл. Для этого необходимо определить закон перестановки символов открытого текста. Создайте небольшой файл длиной в несколько слов с известным вам текстом, зашифруйте его, просмотрите гистограммы (опишите их; ответьте, можно ли извлечь из них полезную для дешифрации информацию). Сравните (с помощью редактора) ваш исходный и зашифрованный тексты и определите закон перестановки символов.

Дешифруйте файл:

- вручную (объясните ваши действия);
- с помощью программы.

5. Для инверсного кодирования (по дополнению до 255):

- выполните шифрование для своего произвольного файла;
- просмотрите гистограммы исходного и зашифрованного текстов, опишите гистограммы и определите смещение для нескольких символов;
- дешифруйте зашифрованный текст, проверьте в редакторе правильность дешифрования.

6. Для многоалфавитного шифрования с фиксированным ключом определите, сколько одноалфавитных методов и с каким смещением

используется в программе. Для этого нужно создать свой файл, состоящий из строки одинаковых символов, выполнить для него шифрование и по гистограмме определить способ шифрования и набор смещений.

7. Для многоалфавитного шифрования с ключом фиксированной длины:

■ выполните шифрование и определите по гистограмме, какое смещение получает каждый символ для файла, состоящего из строки одинаковых символов;

■ выполните шифрование и расшифрование для файла произвольного текста;

■ просмотрите и опишите гистограммы исходного и зашифрованного текстов; ответьте, какую информацию можно получить из гистограмм.

8. Для многоалфавитного шифрования с произвольным паролем задание полностью аналогично п. 7.

9. Привести в отчете ответы на контрольные вопросы в соответствии с номером варианта, указанным преподавателем (табл. 1.1).

Таблица 1.1

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Какие вы знаете методы криптографической защиты файлов?
2, 4, 6, 8, 20, 22, 24, 26, 30	В чем преимущества и недостатки одноалфавитных методов?
11, 13, 15, 10, 17, 19, 27	Если необходимо зашифровать текст, содержащий важную информацию, какой метод из рассмотренных вы выберете? Обоснуйте свой выбор
12, 14, 16, 21, 23, 25, 29	Целесообразно ли повторно применять для уже зашифрованного текста: а) метод многоалфавитного шифрования; б) метод Цезаря?

ЛАБОРАТОРНАЯ РАБОТА № 2

ИССЛЕДОВАНИЕ РАЗЛИЧНЫХ МЕТОДОВ ЗАЩИТЫ ТЕКСТОВОЙ ИНФОРМАЦИИ И ИХ СТОЙКОСТИ НА ОСНОВЕ ПОДБОРА КЛЮЧЕЙ

Цель работы: изучение методов шифрования (расшифрования) перестановкой символов, подстановкой, гаммированием, использованием таблицы Виженера. Исследование и сравнение стойкости различных методов на основе атак путем перебора всех возможных ключей.

В лабораторной работе рассматривается способ вскрытия шифра, основанный на переборе всех вариантов ключа. Критерием правильности варианта служит наличие в тексте «вероятного слова». Перебирается множество всех возможных ключей, шифрованный текст расшифровывается на каждом ключе. В получившемся «псевдооткрытом» тексте ищется вероятное слово. Если такого слова нет, текущий текст бракуется и осуществляется переход к следующему ключу. Если такое слово найдено, на экран выводится вариант ключа. Затем перебор ключей продолжается до тех пор, пока не исчерпается все множество вариантов. Возможно обнаружение нескольких ключей, при которых в «псевдооткрытых текстах» имеется вероятное слово.

После завершения перебора необходимо расшифровать текст на найденных ключах. «Псевдооткрытый текст» выводится на экран для визуального контроля. Если оператор признает текст открытым, то работа по вскрытию заканчивается. Иначе данный вариант ключа бракуется и осуществляется переход к следующему ключу.

Описание лабораторной работы. Для выполнения лабораторной работы необходимо запустить программу **LAB_RAB.exe**, используемую для шифрования (расшифрования), а также дешифрования (методом протяжки вероятного слова) файлов.

Система реализует следующие функции:

- ввод, удаление и селекция ключей пользователя;
- поддержка списка ключей;
- шифрование и расшифрование текста;
- дешифрование текста путем подбора ключей, методом протяжки вероятного слова.

Система поддерживает следующие методы криптографического преобразования информации:

- замена;
- перестановка;

- гаммирование;
- таблица Виженера.

При запуске утилит шифрования и расшифрования у пользователя запрашивается подтверждение на правильность выбранного метода для работы и соответствия заданного ключа целям пользователя (также всегда при изменении файла в текстовом редакторе выдается запрос на сохранение изменений при каждом шаге, дальнейшее развитие которого приведет к необратимым изменениям в файле и потере изначальной информации).

Описание интерфейса:

- окно текстового редактора с широким набором дополнительных функций;
- таблица всех ключей, введенных в систему с быстрым доступом для ввода, удаления или выбора текущего ключа;
- список всех методов шифрования для быстрого и удобного переключения между ними;
- основное меню (вверху экрана);
- дополнительное меню (вызывается нажатием правой кнопки мыши);
- набор вспомогательных кнопок для быстрого и удобного интерфейса;
- поля вывода текущего состояния системы:
  - текущий ключ,
  - вероятное слово,
  - сила ключа для протяжки.

Пример работы с программой

Внимание! Будьте внимательны при установке параметров работы, так как в процессе вычисления по ходу работы эти параметры изменить уже не удастся. После запуска программы абсолютно все рабочие поля пустые и необходимо провести первоначальные настройки для работоспособности системы.

Алгоритм работы с программой:

1) вводится список ключей;
2) вводится вероятное слово (необязательно вначале, до его ввода все меню запуска протяжки все равно недоступны);
3) выбирается необходимый метод шифрования;
4) загружается исходный или зашифрованный файл (открываются соответствующие меню для шифрования и расшифрования);
5) запускается необходимый процесс:
   - шифрование,

— расшифрование,
— протяжка вероятного слова,
— конвертация текста;

6) продолжение работы в любом порядке в соответствии с описанными пунктами;
7) при завершении работы не забудьте сохранить необходимые результаты (при закрытии и загрузке новых файлов система автоматически запрашивает подтверждение на запись).

Шифрование.

1. Открыть файл.
2. Внести необходимые изменения.
3. Настроить соответствующие параметры:

- тип шифрования;
- ключ, пр.

4. Запустить процесс шифрования через пункт меню УТИЛИТЫ/ЗАШИФРОВАТЬ F5.

Внимание! При шифровании файла все внесенные пользователем изменения до текущего момента времени будут сохранены на жестком диске.

Расшифрование.

1. Открыть файл.
2. Произвести необходимые изменения.
3. Настроить соответствующие параметры: тип шифрования, пр.
4. Запустить процесс расшифрования через пункт меню УТИЛИТЫ/РАСШИФРОВАТЬ F6.

Внимание! При расшифровании файла все проведенные пользователем изменения до текущего момента времени будут сохранены на жестком диске.

Протяжка вероятного слова (дешифрование)

Внимание! Мощность ключа задастся заранее в опции «сила ключа». Длина ключа значительно влияет на время протяжки вероятного слова (в худшем случае имеем дело с логарифмическим алгоритмом).

1. Вводится вероятное слово (длиной от 1 (3) до 9).
2. Для отделения вновь найденных ключей от предыдущих между ними добавляется надпись «подбор».
3. Перебираются ключи.
4. Расшифровывается вся первая строка текста по текущему ключу.
5. Порциями, равными длине вероятного слова, сравнивается содержимое этой строки со значением вероятного слова.
6. Если найдено хоть одно совпадение, запоминается ключ.

7. Переход к новому ключу.
8. Переход к следующей строке.
9. Результаты должны содержаться в списке ключей. Если совпадений не найдено, в список ключей ничего не добавляется.

Операции с ключами. С базой ключей могут осуществляться следующие действия:

- добавить новый ключ;
- удалить одну запись;
- изменить активную запись;
- очистить всю таблицу введенных ключей.

Примечание. Под словами «работа с таблицей ключей» имеются в виду ключи, введенные для использования в двух методах (гаммирования и таблица Виженера).

Ключи для перестановки. В каждый момент времени в системе может быть только один текущий ключ для перестановки.

Правила ввода ключа для перестановки:

1) при переключении в списке поддерживаемых системой методов шифрования на пункт «Перестановка» вызывается окно ввода ключа перестановки. Окно состоит из двух кнопок (ОТМЕНЫ и ВЫХОДА без изменений и кнопки ENTER — подтверждение установленной длины ключа) и окна задания длины ключа для перестановки;
2) в окне задания длины ключа необходимо выбрать необходимую длину (параметры задаются в пределах 1...9) и подтвердить желание использовать ключ именно такой длины;
3) после подтверждения в окне высветятся кнопки с цифрами на лицевой стороне (в количестве, равном длине ключа), при нажатии на кнопку происходит фиксация кнопки (ее обесцвечивание) для невозможности ее дальнейшего использования (так как все цифры в ключе перестановки должны быть неповторяющимися);
4) после перебора всех кнопок система запоминает введенный ключ, выводит его в поле ввода ключей и выходит из окна ввода ключа перестановки в окно основной программы.

Задание

1. Ознакомиться с описанием лабораторной работы и заданием.
2. Выполнить настройку программы: выбрать метод шифрования, ввести ключи для всех методов, ввести вероятное слово, осуществить все остальные системные настройки.

3. Для метода замены (одноалфавитного метода):

■ выбрать данный алгоритм в списке доступных методов шифрования;

■ установить необходимое смещение;

■ открыть произвольный файл;

■ просмотреть содержимое исходного файла;

■ выполнить для этого файла шифрование (при необходимости можно задать имя зашифрованного файла);

■ просмотреть в редакторе зашифрованный файл;

■ ввести вероятное слово;

■ ввести вероятную длину ключа (кроме метода замены);

■ подобрать ключ;

■ выполнить расшифрование со всеми найденными ключами;

■ найти в каком-либо из расшифрованных файлов правильно расшифрованное ключевое слово;

■ расшифровать файл исходным ключом;

■ проверить результат.

4. Для метода перестановки:

■ выбрать метод перестановки;

■ в открывшемся окне ввода ключа перестановки символов указать сначала длину этого ключа, а затем из появившихся кнопок составить необходимую комбинацию для ключа, нажимая на кнопки в заданном порядке; при этом уже использованные кнопки становятся недоступными для предотвращения их повторного ввода;

■ далее действия полностью соответствуют изложенным в п. 3.

5. Для метода гаммирования:

■ выбрать метод;

■ ввести ключ;

■ полностью повторить п. 3.

6. Для таблицы Виженера все действия повторяются из п. 5 (метод гаммирования).

В отчете для каждого метода шифрования описывается последовательность выполняемых действий, указываются имена всех использованных файлов, исходные и найденные ключи, описывается процесс дешифрования.

Преподавателю предоставляется отчет о проделанной работе и все использованные файлы.

7. Привести в отчете ответы на контрольные вопросы в соответствии с номером варианта, указанным преподавателем (табл. 1.2).

Таблица 1.2

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Чем отличается псевдооткрытый текст (текст, полученный при расшифровке по ложному ключу) от настоящего открытого текста?
2, 4, 6, 8, 20, 22, 24, 26, 30	Как зависит время вскрытия шифра описанным выше способом подбора ключей от длины вероятного слова?
11, 13, 15, 10, 17, 19, 27	Зависит ли время вскрытия шифра гаммирования (или таблицы Виженера) от мощности алфавита гаммы?
12, 14, 16, 21, 23, 25, 29	В чем недостатки метода дешифрования с использованием протяжки вероятного слова?

ЛАБОРАТОРНАЯ РАБОТА № 3

ИЗУЧЕНИЕ УСТРОЙСТВА И ПРИНЦИПА РАБОТЫ ШИФРОВАЛЬНОЙ МАШИНЫ «ЭНИГМА»

Цель работы: изучение принципов шифрования (расшифрования) информации, используемых в шифровальной машине «Энигма». Ознакомление с общими принципами действия шифровальной машины «Энигма» на примере эмулятора. Предварительно необходимо установить программу-эмулятор Enigma.

Описание лабораторной работы. «Энигма» (*Enigma*) — портативная шифровальная машина, использовавшаяся для шифрования и расшифрования секретных сообщений (рис. 1.9). Более точно, «Энигма» — целое семейство электромеханических роторных машин, применявшихся с 1920-х гг. «Энигма» использовалась в коммерческих целях, а также в военных и государственных службах во многих странах мира, но наибольшее распространение получила в Германии во время Второй мировой войны. Именно «Энигма» Вермахта (Wehrmacht Enigma), немецкая военная модель, чаще всего является предметом изучения.

Рис. 1.9. Внешний вид шифровальной машины «Энигма»

Хотя шифр «Энигмы» с точки зрения криптографии был достаточно слаб, но на практике лишь сочетание этого фактора с другими, такими как ошибки операторов, процедурные изъяны и захваты экземпляров «Энигмы» и шифровальных книг, позволило английским криптоаналитикам вскрывать сообщения, зашифрованные шифром «Энигмы».

На рисунке 1.10 показана электрическая схема машины «Энигма» для двух последовательно нажатых клавиш — ток течет через роторы, «отражается» от рефлектора, затем снова возвращается через роторы. Буква «A» заменяется в шифротексте по-разному при последовательных нажатиях клавиши, сначала на «G», затем на «C». Сигнал идет по другому маршруту за счет поворота ротора.

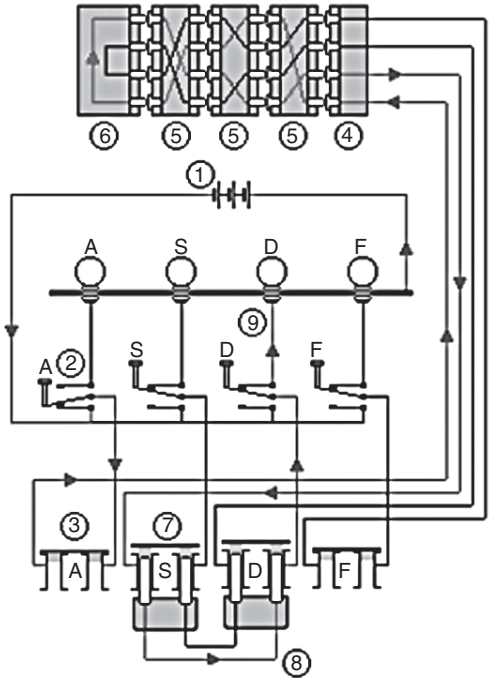


Рис. 1.10. Электрическая схема машины «Энигма» (замена в тексте буквы «A» буквой «D»)

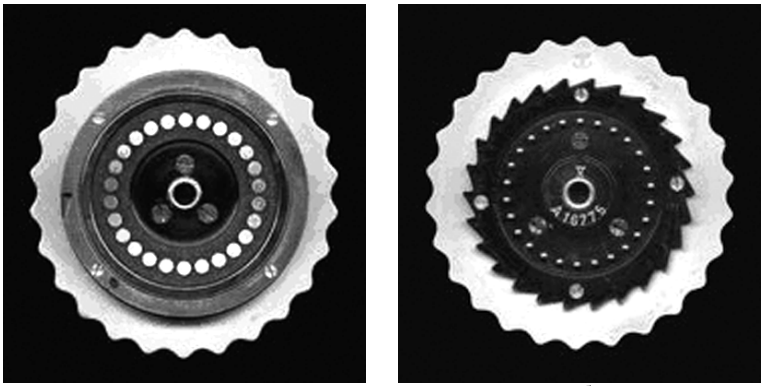
Как и другие роторные машины, «Энигма» состояла из комбинации механических и электрических систем. Механическая часть включала клавиатуру, набор вращающихся дисков (роторов), расположенных вдоль вала, и ступенчатого механизма, обеспечивающего движение одной или более роторов при каждом нажатии клавиши. Движение роторов приводит к различным вариантам подстановки символов при каждом следующем нажатии клавиши на клавиатуре.

Механические части двигались, образуя меняющийся электрический контур, т.е. фактически шифрование осуществлялось электри-

чески. При нажатии клавиш контур замыкался, ток проходил через различные компоненты и в итоге включал одну из множества лампочек, отображавшую выводимую букву. Например, при шифровании сообщения, начинающегося ANX..., оператор вначале нажимал кнопку «A», и загоралась лампочка «Z», т.е. «Z» становилась первой буквой криптограммы. Оператор продолжал шифрование, нажимая на клавиатуре «N» и т.д., до конца исходного сообщения.

Постоянное изменение электрической цепи, через которую протекал ток, вследствие вращения роторов позволяло реализовать многоалфавитный шифр подстановки, что давало высокую стойкость шифрования для того времени.

Роторы. Роторы — это сердце машины «Энигмы» (рис. 1.11). Каждый ротор представляет собой диск примерно 10 см в диаметре, сделанный из твердой резины или бакелита, с пружинными штыревыми контактами на одной стороне ротора, расположенными по окружности; на другой стороне ротора находится соответствующее количество плоских электрических контактов. Штыревые и плоские контакты соответствуют буквам в алфавите; обычно это 26 букв «A» — «Z». При соприкосновении контакты соседних роторов замыкают электрическую цепь. Внутри ротора каждый штыревой контакт соединен с некоторым плоским. Порядок соединения может быть различным.



а *б*

Рис. 1.11. Ротор машины «Энигмы»: *а* — левая сторона(видны плоские электрические контакты); *б* — правая сторона (видны штыревые контакты)

Сам по себе ротор воспроизводит шифрование простой заменой символов. Например, контакт, отвечающий за букву «E», может быть соединен с контактом буквы «T» на другой стороне ротора. При ис-

пользовании нескольких роторов в связке (обычно трех или четырех) за счет их постоянного движения получается более стойкий тип многоалфавитного шифрования.

Ротор может занимать одну из 26 позиций в машине. Он может быть повернут вручную при помощи рифленого пальцевого колесика, которое выдается наружу, как показано на рис. 1.12. Чтобы оператор всегда мог определить положение ротора, на каждом ободе находится алфавитное кольцо; одна из букв видна через окошко. В ранних моделях «Энигмы» алфавитное кольцо было фиксировано, в более поздних версиях ввели усложненную конструкцию с возможностью его регулировки. Каждый ротор содержит выемку (или несколько выемок), используемых для управления движением роторов. Три последовательно соединенных ротора изображены на рис 1.13.

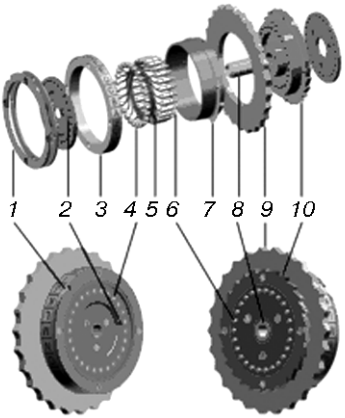


Рис. 1.12. Ротор в разобранном виде: *1* — кольцо с выемками; *2* — маркирующая точка для контакта «А»; *3* —алфавитное кольцо; *4* — залуженные контакты; *5* — электропроводка; *6* — штыревые контакты; *7* — пружинный рычаг для настройки кольца; *8* — втулка; *9* —пальцевое кольцо; *10* —храповое колесо

Военные версии машины «Энигма» выпускались с несколькими роторами; первая модель содержала только три. В 1938 году их стало пять, но только три из них одновременно использовались в машине. Эти типы роторов были маркированы римскими цифрами I, II, III, IV, V, и все они были с одной выемкой, расположенной в разных местах алфавитного кольца. В военно-морских версиях Wehrmacht Enigma содержалось большее количество роторов, чем в других: шесть, семь или восемь.

Рис. 1.13. Три последовательно соединенных ротора

В *Wehrmacht Enigma* каждый ротор прикреплен к регулируемому кольцу с выемками. Пять базовых роторов (I—V) имели по одной выемке, тогда как военно-морские с дополнительными роторами (VI—VIII) — по две. В определенный момент выемка попадает напротив, собачки, позволяя и зацепить храповик следующего ротора при последующем нажатии клавиши. Когда же собачка не попадает в выемку, она просто проскальзывает по поверхности кольца, не цепляя шестеренки. В системе с одной выемкой второй ротор продвигается вперед на одну позицию в то время, как первый продвигается на 26. Аналогично третий ротор продвигается на один шаг в то время, как второй делает 26 шагов. Особенностью было то, что второй ротор также поворачивался, если поворачивался третий; это означает, что второй ротор мог повернуться дважды при двух последовательных нажатиях клавиш, так называемое «двухшаговое движение», что приводит к уменьшению периода при шифровании (рис. 1.14).

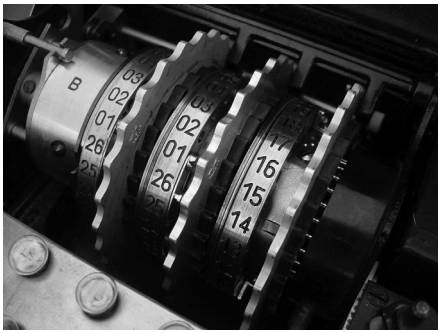


Рис. 1.14. Роторы энигмы в собранном состоянии. Три подвижных ротора помещены между двумя неподвижными деталями: входное кольцо и рефлектор (помечен «B» слева)

Входное колесо. Входное колесо (нем. Eintrittswalze) машины «Энигма», или входной статор, соединяет коммутационную панель или (в случае ее отсутствия) клавиатуру и ламповую панель с роторами. Коммерческая версия «Энигмы» соединяла буквы в порядке их следования на клавиатуре: QA, WB, EC и т.д. Однако в военной модели они соединялись в прямом алфавитном порядке: AA, BB, CC и т.д.

Рефлектор. За исключением ранних, за последним ротором устанавливался рефлектор (нем. Umkehrwalze) — запатентованная деталь, отличавшая семейство машин «Энигма» от других роторных машин, разработанных в то время. Рефлектор соединяет контакты последнего ротора попарно, коммутируя ток через роторы в обратном направлении, но по другому маршруту. Рефлектор гарантирует, что преобразование, реализуемое машиной, инволюция, т.е. процесс расшифрования, симметричен процессу шифрования. Кроме того, рефлектор гарантирует, что никакая буква не может быть зашифрована собой же. Это оказалось серьезным концептуальным недостатком, впоследствии использованным дешифровальщиками.

Коммутационная панель. Коммутационная панель, позволяющая оператору варьировать соединения проводов, впервые появилась в немецких военных моделях в 1930 г. и вскоре успешно использовалась и в машинах для военно-морских войск (рис. 1.15). Коммутационная панель внесла огромный вклад в усложнение шифра «Энигмы», даже больший, чем введение дополнительного ротора. С «Энигмой» без коммутационной панели дешифровальщик может справиться практически вручную, однако при использовании коммутационной панели взломщики были вынуждены конструировать специальные дешифровальные машины. Кабель, помещенный на коммутационную панель, соединяет буквы попарно, например «E» и «Q» могут быть соединены в пару. Эффект состоит в перестановке этих букв до и после

Рис. 1.15. Коммутационная панель «Энигмы»

прохождения сигнала через роторы. Например, когда оператор нажимал клавишу буквы «E», сигнал направлялся в «Q» и только после этого уже во входной ротор. Одновременно могло использоваться несколько таких пар (до 13).

Аксессуары. Удобной деталью, установленной на модели M4 «Энигма», был «*Schreibmax*» — маленькое печатающие устройство, которое могло печатать все 26 букв на небольшом листе бумаги.

Процедуры использования «Энигмы». Во время Второй мировой войны немецкие операторы использовали шифровальную книгу только для установки роторов и настроек колец. Для каждого сообщения они выбирали случайную стартовую позицию, например WZA, и случайный ключ сообщения, например SXT. Затем оператор устанавливал роторы в стартовую позицию WZA и шифровал ключ сообщения SXT. Предположим, что в результате шифрования ключа получится UHL. Далее оператор ставил ключ сообщения SXT как начальную позицию роторов и шифровал сообщение. После чего отправлял стартовую позицию WZA и зашифрованный ключ UHL вместе с сообщением. Получатель устанавливал стартовую позицию в соответствии с первой трехграммой WZA и, расшифровывая вторую триграмму UHL, распознавал исходный ключ SXT. Далее получатель использовал этот ключ как стартовую позицию для расшифровки сообщения. Обычно срок действия ключей составлял один день.

Военная модель «Энигма» использовала только 26 букв. Прочие символы заменялись редкими комбинациями букв. Пробел пропускался либо заменялся «X». Символ «X» также использовался для обозначения точки либо конца сообщения. Некоторые особые символы использовались в отдельных вооруженных частях, например, Wehrmacht заменял запятую двумя символами ZZ и вопросительный знак — FRAGE либо FRAQ, а Kriegsmarine заменяла запятую — «Y» и вопросительный знак — UD. Два, три или четыре нуля заменялись CENTA, MILLE и MYRIA соответственно.

При выполнении лабораторной работы для исследования шифра «Энигмы» используется программа-эмулятор Enigma3S.

Задание

1. Запустить эмулятор Энигмы Enigma из папки, указанной преподавателем. Ознакомиться с файлом справки: опция меню Help (рис. 1.16).
2. 2. Открыть настройки Энигмы нажатием «Open cover» (рис. 1.17).

Рис. 1.16. Меню открывается нажатием красной кнопки

Рис. 1.17. Просмотр начальных установок

3. Установить значение рефлектора в положение B. Значение рефлектора меняется кликом по нему (рис. 1.18)

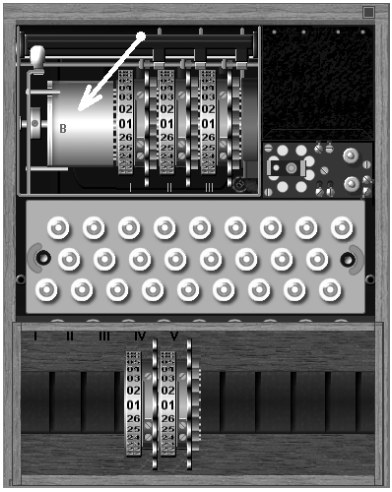


Рис. 1.18. Рефлектор

4. Установить настройки роторов L-I, M-II, R-III.

4.1. Для смены настроек ротора необходимо нажать на ротор, который необходимо заменить (рис. 1.19).

4.2. Затем нажать на бокс для роторов, ротор переместится в бокс для роторов (рис. 1.20).

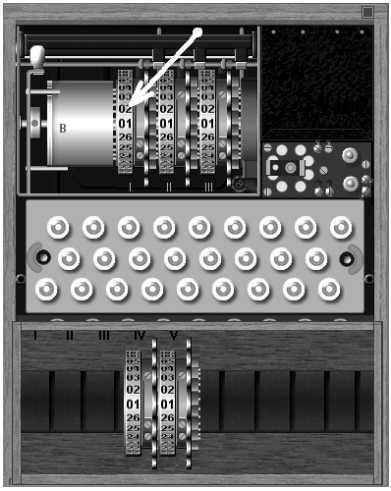


Рис. 1.19. Выбор ротора, который необходимо заменить

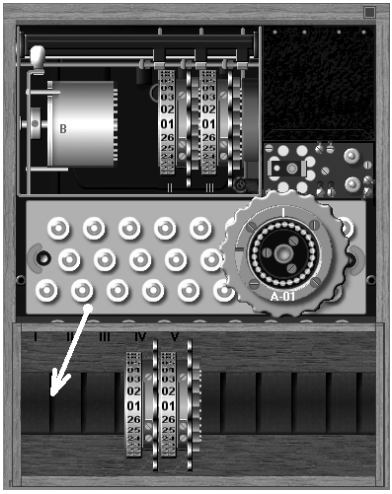


Рис. 1.20. Поместить выбранный ротор в бокс для роторов

4.3. Выбрать из бокса необходимый ротор (рис. 1.21).

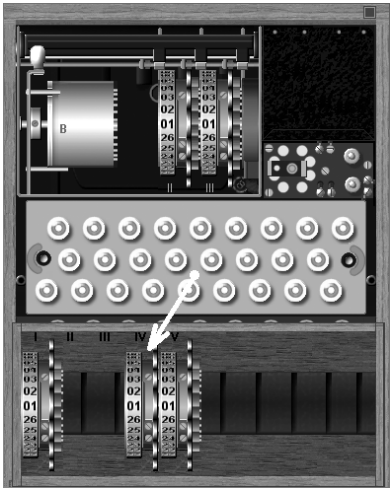


Рис. 1.21. Выбор нужного ротора

4.4. Нажав на позицию, установить ротор (рис. 1.22).

4.5. После выполнения необходимых настроек можно закрыть меню настроек нажатием «Close Cover» (рис. 1.23).

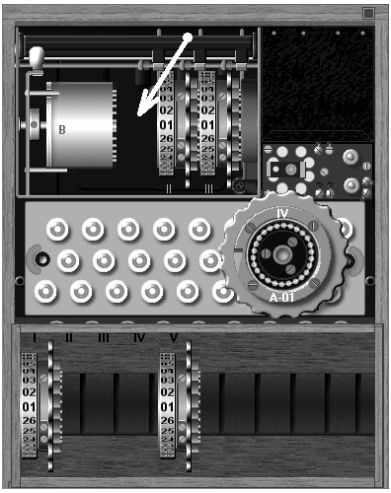


Рис. 1.22. Установка ротора на позицию

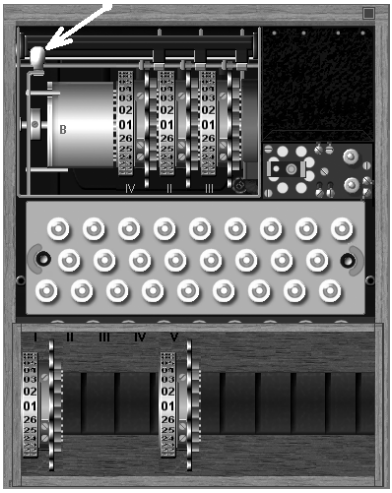


Рис. 1.23. Закрытие меню настроек

5. Установить начальные значения роторов в позиции AAA (111). Настройка регулируется нажатием на шестеренку ротора (рис. 1.24).

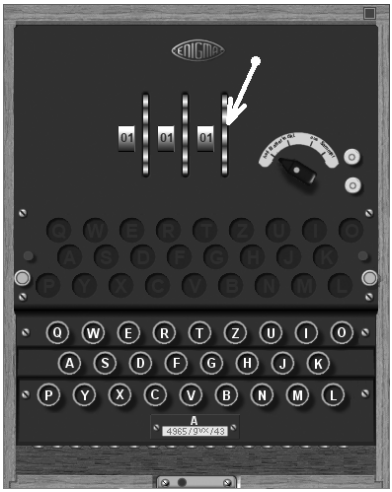


Рис. 1.24. Смена начальных значений роторов

Настройки Reflector-B, L-I M-II R- III, AAA считать начальными.

6. Нажать «Hide Textbox» (рис. 1.25).

Рис. 1.25. Hide Textbox

7. Ввести на клавиатуре «Энигма» сообщение SECRET MESSAGE. Какое сообщение получено на выходе?

8. Изменить настройки Ringstellung — A-A-A на Ringstellung — A-B-C. Ввести на клавиатуре «Энигма» сообщение SECRET MESSAGE. Какое сообщение получено на выходе? На сколько оно отличается от сообщения, полученного в п. 7?

9. Открыть меню, выбрать «Auto Typing», в открывшемся окне ввести либо вставить текст и нажать «Start», получить шифротекст (рис. 1.26).

10. Установить эмулятор в начальное положение. Ввести произвольное сообщение из 22-х символов (варианты указаны в табл. 1.3), обращая внимание на положение колец. Ввести последовательность из 22-х символов еще раз. Как изменилось положение колец? Отличается ли новая зашифрованная последовательность от начальной? Почему изменилась выходная последовательность?

11. Проследить за тем, как изменяется шифротекст в зависимости от настройки положения контактных колес. Что дает возможность настройки порядка следования контактных колес?

12. Заполнить таблицу соответствия для пяти букв (варианты указаны в табл. 1.4) для 12-и первых угловых положений правого колеса.

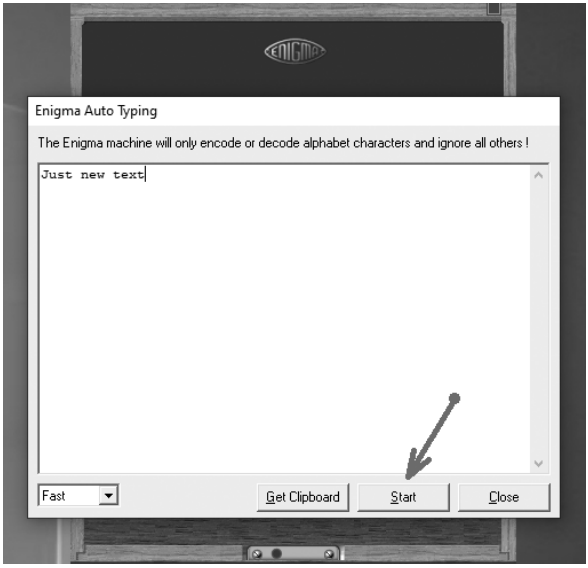


Рис. 1.26. Автоввод текста

Таблица 1.3

Номер варианта	Исходный алфавит
1, 5, 9, 13, 17	QRTYU FBNAK GHERL ADLKE DS
2, 6, 10, 14, 18, 22	UIERT PAEVC DSNCY OPLKD BV
3, 7, 11, 15, 19, 23, 27	UIFGH KLBVQ FDIIT QKJLS DB
4, 8, 12, 16, 20, 24, 28	LLWER TYYWV BAFDP WRTOPF JK
21, 25, 29, 26, 30	OPJHG JFDPJ GFDSK LDFHU BX

Таблица 1.4

Буквы на входе	Угловые положения колеса											
	1	2	3	4	5	6	7	8	9	10	11	12

13. Зная, что эмулятор установлен в начальное положение, расшифровать текст в соответствии с номером варианта, указанным преподавателем (варианты контрольных заданий указаны в табл. 1.5, 1.6.)

Таблица 1.5

Номер варианта	Шифротекст
1, 5, 9, 13, 17	ABCDE
2, 6, 10, 14, 18, 22	FGHIJK
3, 7, 11, 15, 19, 23, 27	LMNOP
4, 8, 12, 16, 20, 24, 28	QRSTU
21, 25, 29, 26, 30	VWXYZ

Таблица 1.6

Номер варианта	Шифротекст
1, 5, 9, 13, 17	FQGAH WABUN NL
2, 6, 10, 14, 18, 22	QIKOL RCRJS EGBSS X
3, 7, 11, 15, 19, 23, 27	OOKWE PRFMI M
4, 8, 12, 16, 20, 24, 28	KIXDI ACTHJ L
21, 25, 29, 26, 30	XLXOO EABUN NL

ЛАБОРАТОРНАЯ РАБОТА № 4

СТАНДАРТ СИММЕТРИЧНОГО ШИФРОВАНИЯ AES RIJNDAEL

Цель работы: ознакомление с принципами шифрования, используемыми в алгоритме симметричного шифрования AES RIJNDAEL.

Описание лабораторной работы. Для выполнения работы используется демонстрационная версия криптостойкого блочного алгоритма Rijndael. Rijndael — итеративный блочный шифр, имеющий переменную длину блоков и различные длины ключей. Длина ключа и длина блока могут независимо друг от друга составлять 128, 192 или 256 бит.

Разнообразные преобразования работают с промежуточным результатом, называемым состоянием (State). Состояние можно представить в виде прямоугольного массива байтов. Этот массив имеет четыре строки, а число столбцов обозначается Nb и равно длине блока, деленной на 32.

Ключ шифрования также представлен в виде прямоугольного массива с четырьмя строками. Число столбцов обозначено как Nk и равно длине ключа, деленной на 32 (рис. 1.27).

В некоторых случаях ключ шифрования изображается в виде линейного массива четырехбайтовых слов. Слова состоят из четырех байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Рис. 1.27. Пример представления состояния (Nb = 6) и ключа шифрования (Nk = 4)

Входные данные для шифра обозначаются как байты состояния в порядке $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$, ... После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке.

Число циклов, обозначенное Nr, зависит от значений Nb и Nk (табл. 1.7).

Таблица 1.7

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Цикловое преобразование. Цикловое преобразование состоит из четырех различных преобразований. На языке псевдо-Си оно имеет следующий вид:

```
Round (State, RoundKey)
{
     ByteSub(State); // замена байт
     ShiftRow(State); // сдвиг строк
     MixColumn(State); // замешивание столбцов
     AddRoundKey(State, RoundKey); // добавление ци-
клового ключа
}
```

Последний цикл шифра немного отличается:

```
FinalRound(State, RoundKey)
{
     ByteSub(State); // замена байт
     ShiftRow(State); // сдвиг строк
     AddRoundKey(State, RoundKey); // добавление цикло-
вого ключа
}
```

Отметим, что последний цикл отличается от простого цикла только отсутствием замешивания столбцов. Каждое из приведенных преобразований подробно рассмотрено далее.

Замена байт (*ByteSub*). Преобразование ByteSub — нелинейная замена байт, выполняемая независимо с каждым байтом состояния (рис. 1.28).

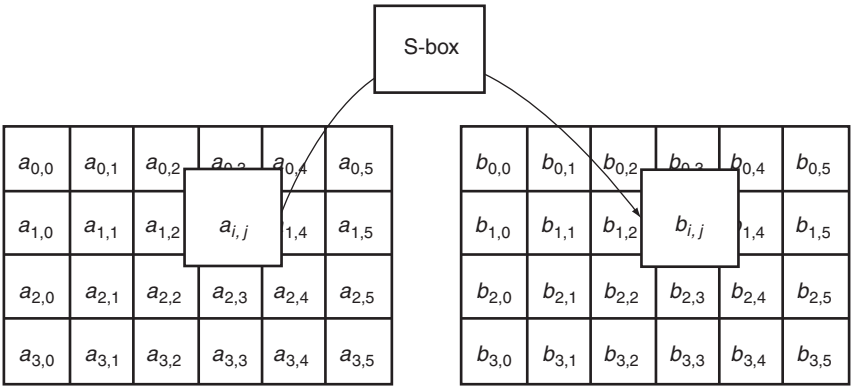


Рис. 1.28. ByteSub действует на каждый байт состояния

Замена происходит по массиву SboxE при шифровании и по массиву SboxD при расшифровании, причем SboxD[SboxE[a]] = a. На языке псевдо-Си это выглядит следующим облазом:

```
SboxE = {
0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30,
0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD,
0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34,
0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07,
0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52,
0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84, 0x53, 0xD1,
0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE,
0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF, 0xAA, 0xFB,
0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50,
0x3C, 0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D,
0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3,
0xD2, 0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17,
0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73, 0x60,
0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE,
0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB, 0xE0, 0x32, 0x3A,
0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62,
0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D, 0x8D,
```

```
0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A,
0xAE, 0x08,
0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8,
0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61,
0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B,
0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41,
0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
};

SboxD = {
0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF,
0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34,
0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE,
0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76,
0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4,
0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E,
0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7,
0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1,
0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97,
0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2,
0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F,
0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A,
0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1,
0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
```

```
0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D,
0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8,
0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1,
0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
};
```

Преобразование сдвига строк (*ShiftRow*). Последние три строки состояния циклически сдвигаются на различное число байт. Строка 1 сдвигается на C1 байт, строка 2 — на C2 байт и строка 3 — на C3 байт. Значения сдвигов C1, C2 и C3 зависят от длины блока Nb. Их величины приведены в табл. 1.8.

Таблица 1.8

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Операция сдвига последних трех строк состояния на определенную величину обозначена ShiftRow (State). На рисунке 1.29 показано влияние преобразования на состояние.

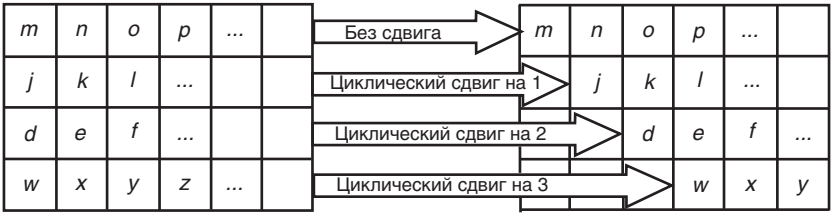


Рис. 1.29. Схема преобразования ShiftRow

При расшифровании происходит сдвиг на то же число элементов в обратном направлении.

Преобразование замешивания столбцов (MixColumn). Преобразование представляет собой умножение состояния на матрицу ME при шифровании или матрицу MD при расшифровании:

			ME	⊗	State			
			MD	⊗	State			
2	3	1	1	⊗	b1	b5	b9	b13
1	2	3	1		b2	b6	b10	b14
1	1	2	3		b3	b7	b11	b15
3	1	1	2		b4	b8	b12	b16

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
```

Умножение двух байт выполняется по следующему алгоритму:

- если один из байт равен 0, результатом будет 0;
- если один из байт равен 1, результатом будет другой байт;
- в остальных случаях происходит замена каждого байта по таблице L. Замененные байты складываются, при необходимости вычитается 255 для попадания в интервал [0, 255] и происходит замена по таблице E, что и дает результат. На языке псевдо-Си таблицы L и E имеют следующий вид:

```
L = {
    , 0x00, 0x19, 0x01, 0x32, 0x02, 0x1A, 0xC6, 0x4B,
0xC7, 0x1B, 0x68, 0x33, 0xEE, 0xDF, 0x03,
0x64, 0x04, 0xE0, 0x0E, 0x34, 0x8D, 0x81, 0xEF, 0x4C,
0x71, 0x08, 0xC8, 0xF8, 0x69, 0x1C, 0xC1,
0x7D, 0xC2, 0x1D, 0xB5, 0xF9, 0xB9, 0x27, 0x6A, 0x4D,
0xE4, 0xA6, 0x72, 0x9A, 0xC9, 0x09, 0x78,
0x65, 0x2F, 0x8A, 0x05, 0x21, 0x0F, 0xE1, 0x24, 0x12,
0xF0, 0x82, 0x45, 0x35, 0x93, 0xDA, 0x8E,
0x96, 0x8F, 0xDB, 0xBD, 0x36, 0xD0, 0xCE, 0x94, 0x13,
0x5C, 0xD2, 0xF1, 0x40, 0x46, 0x83, 0x38,
0x66, 0xDD, 0xFD, 0x30, 0xBF, 0x06, 0x8B, 0x62, 0xB3,
0x25, 0xE2, 0x98, 0x22, 0x88, 0x91, 0x10,
0x7E, 0x6E, 0x48, 0xC3, 0xA3, 0xB6, 0x1E, 0x42, 0x3A,
0x6B, 0x28, 0x54, 0xFA, 0x85, 0x3D, 0xBA,
0x2B, 0x79, 0x0A, 0x15, 0x9B, 0x9F, 0x5E, 0xCA, 0x4E,
0xD4, 0xAC, 0xE5, 0xF3, 0x73, 0xA7, 0x57,
0xAF, 0x58, 0xA8, 0x50, 0xF4, 0xEA, 0xD6, 0x74, 0x4F,
0xAE, 0xE9, 0xD5, 0xE7, 0xE6, 0xAD, 0xE8,
0x2C, 0xD7, 0x75, 0x7A, 0xEB, 0x16, 0x0B, 0xF5, 0x59,
0xCB, 0x5F, 0xB0, 0x9C, 0xA9, 0x51, 0xA0,
```

```
0x7F, 0x0C, 0xF6, 0x6F, 0x17, 0xC4, 0x49, 0xEC, 0xD8,
0x43, 0x1F, 0x2D, 0xA4, 0x76, 0x7B, 0xB7,
0xCC, 0xBB, 0x3E, 0x5A, 0xFB, 0x60, 0xB1, 0x86, 0x3B,
0x52, 0xA1, 0x6C, 0xAA, 0x55, 0x29, 0x9D,
0x97, 0xB2, 0x87, 0x90, 0x61, 0xBE, 0xDC, 0xFC, 0xBC,
0x95, 0xCF, 0xCD, 0x37, 0x3F, 0x5B, 0xD1,
0x53, 0x39, 0x84, 0x3C, 0x41, 0xA2, 0x6D, 0x47, 0x14,
0x2A, 0x9E, 0x5D, 0x56, 0xF2, 0xD3, 0xAB,
0x44, 0x11, 0x92, 0xD9, 0x23, 0x20, 0x2E, 0x89, 0xB4,
0x7C, 0xB8, 0x26, 0x77, 0x99, 0xE3, 0xA5,
0x67, 0x4A, 0xED, 0xDE, 0xC5, 0x31, 0xFE, 0x18, 0x0D,
0x63, 0x8C, 0x80, 0xC0, 0xF7, 0x70, 0x07
};

E = {
0x01, 0x03, 0x05, 0x0F, 0x11, 0x33, 0x55, 0xFF, 0x1A,
0x2E, 0x72, 0x96, 0xA1, 0xF8, 0x13, 0x35,
0x5F, 0xE1, 0x38, 0x48, 0xD8, 0x73, 0x95, 0xA4, 0xF7,
0x02, 0x06, 0x0A, 0x1E, 0x22, 0x66, 0xAA,
0xE5, 0x34, 0x5C, 0xE4, 0x37, 0x59, 0xEB, 0x26, 0x6A,
0xBE, 0xD9, 0x70, 0x90, 0xAB, 0xE6, 0x31,
0x53, 0xF5, 0x04, 0x0C, 0x14, 0x3C, 0x44, 0xCC, 0x4F,
0xD1, 0x68, 0xB8, 0xD3, 0x6E, 0xB2, 0xCD,
0x4C, 0xD4, 0x67, 0xA9, 0xE0, 0x3B, 0x4D, 0xD7, 0x62,
0xA6, 0xF1, 0x08, 0x18, 0x28, 0x78, 0x88,
0x83, 0x9E, 0xB9, 0xD0, 0x6B, 0xBD, 0xDC, 0x7F, 0x81,
0x98, 0xB3, 0xCE, 0x49, 0xDB, 0x76, 0x9A,
0xB5, 0xC4, 0x57, 0xF9, 0x10, 0x30, 0x50, 0xF0, 0x0B,
0x1D, 0x27, 0x69, 0xBB, 0xD6, 0x61, 0xA3,
0xFE, 0x19, 0x2B, 0x7D, 0x87, 0x92, 0xAD, 0xEC, 0x2F,
0x71, 0x93, 0xAE, 0xE9, 0x20, 0x60, 0xA0,
0xFB, 0x16, 0x3A, 0x4E, 0xD2, 0x6D, 0xB7, 0xC2, 0x5D,
0xE7, 0x32, 0x56, 0xFA, 0x15, 0x3F, 0x41,
0xC3, 0x5E, 0xE2, 0x3D, 0x47, 0xC9, 0x40, 0xC0, 0x5B,
0xED, 0x2C, 0x74, 0x9C, 0xBF, 0xDA, 0x75,
0x9F, 0xBA, 0xD5, 0x64, 0xAC, 0xEF, 0x2A, 0x7E, 0x82,
0x9D, 0xBC, 0xDF, 0x7A, 0x8E, 0x89, 0x80,
0x9B, 0xB6, 0xC1, 0x58, 0xE8, 0x23, 0x65, 0xAF, 0xEA,
0x25, 0x6F, 0xB1, 0xC8, 0x43, 0xC5, 0x54,
```

```
0xFC, 0x1F, 0x21, 0x63, 0xA5, 0xF4, 0x07, 0x09, 0x1B,
0x2D, 0x77, 0x99, 0xB0, 0xCB, 0x46, 0xCA,
0x45, 0xCF, 0x4A, 0xDE, 0x79, 0x8B, 0x86, 0x91, 0xA8,
0xE3, 0x3E, 0x42, 0xC6, 0x51, 0xF3, 0x0E,
0x12, 0x36, 0x5A, 0xEE, 0x29, 0x7B, 0x8D, 0x8C, 0x8F,
0x8A, 0x85, 0x94, 0xA7, 0xF2, 0x0D, 0x17,
0x39, 0x4B, 0xDD, 0x7C, 0x84, 0x97, 0xA2, 0xFD, 0x1C,
0x24, 0x6C, 0xB4, 0xC7, 0x52, 0xF6, 0x01
};
```

Добавление циклового ключа. Цикловой ключ добавляется к состоянию посредством простого EXOR (рис. 1.30). Цикловой ключ вырабатывается из ключа шифрования посредством алгоритма выработки ключей (key schedule). Длина циклового ключа равна длине блока Nb.

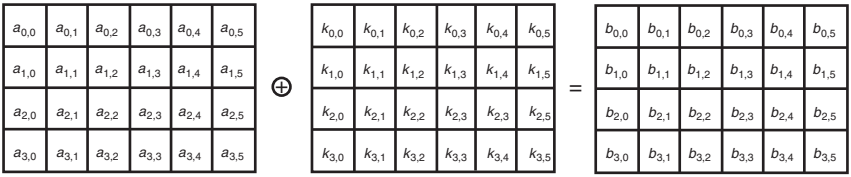


Рис. 1.30. Операция добавления циклового ключа

При шифровании части расширенного ключа выбираются от начала к концу, при расшифровании — от конца к началу.

Расширение ключа (*Key Expansion*). Расширенный ключ представляет собой линейный массив четырех байтовых слов и обозначается W[Nb*(Nr + 1)]. Первые Nk слов содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами. Алгоритм выработки ключей зависит от величины Nk. Ниже приведена версия для Nk ≤ 6 и версия для Nk > 6:

- для Nk < 6 или Nk = 6

```
KeyExpansion(CipherKey,W)
{
     for (i = 0; i < Nk; i++) W[i] = CipherKey[i];
       for (j = Nk; j < Nb*(Nk+1); j+=Nk)
       {
         W[j] = W[j-Nk] ^ SubByte( Rotl( W[j-1] ) ) ^
         Rcon[j/Nk];
         for (i = 1; i < Nk && i+j < Nb*(Nr+1); i++)
```

```
          W[i+j] = W[i+j-Nk] ^ W[i+j-1];
        }
}
```

Можно заметить, что первые Nk слов заполняются ключом шифрования. Каждое последующее слово W[i] получается посредством EXOR предыдущего слова W[i – 1] и слова на Nk позиций ранее W[i – Nk]. Для слов, позиция которых кратна Nk, перед EXOR применяется преобразование к W[i – 1], а затем еще прибавляется цикловая константа. Преобразование содержит циклический сдвиг байтов в слове, обозначенный как Rotl, затем следует SubByte — применение замены байт;

- для Nk > 6

```
KeyExpansion(CipherKey,W)
{
     for (i=0; i<Nk; i++) W[i]=CipherKey[i];
        for (j=Nk; j<Nb*(Nk+1); j+=Nk)
        {
          W[j] = W[j-Nk] ^ SubByte(Rotl(W[j-1])) ^
          Rcon[j/Nk];
          for (i=1; i<4; i++) W[i+j] = W[i+j-Nk] ^
          W[i+j-1];
          W[j+4] = W[j+4-Nk] ^ SubByte(W[j+3]);
          for (i=5; i<Nk; i++) W[i+j] = W[i+j-Nk] ^
          W[i+j-1];
        }
}
```

Отличие для схемы при Nk > 6 состоит в применении SubByte для каждого четвертого байта из Nk.

Цикловая константа независит от Nk и определяется следующим образом:

```
Rcon[i] = ( RC[i], '00' , '00' , '00' ), где
RC[0]='01'
RC[i]=xtime(Rcon[i-1])
```

Шифрование. Шифр Rijndael включает следующие преобразования:

- начальное добавление циклового ключа;
- Nr – 1 циклов;

- заключительный цикл.

На языке псевдо-Си это выглядит следующим образом:

```
Rijndael (State, CipherKey)
{
     KeyExpansion(CipherKey, ExpandedKey); // Расши-
     рение ключа
     AddRoundKey(State, ExpandedKey); // Добавление
     циклового ключа
     For(i=1;i<Nr;i++)Round(State,ExpandedKey+Nb*i);
     // циклы
     FinalRound(State, ExpandedKey+Nb*Nr); // заклю-
     чительный цикл
}
```

Если предварительно выполнена процедура расширения ключа, то процедура будет иметь следующий вид:

```
Rijndael (State, CipherKey)
{
     AddRoundKey(State, ExpandedKey);
     For ( i=1 ; i<Nr ; i++) Round(State, ExpandedKey+Nb*i);
     FinalRound(State, ExpandedKey+Nb*Nr);
}
```

Описание демонстрационной программы. Программа выполнена на языке C# и состоит из двух элементов — файла Rijndael.dll, содержащего реализацию алгоритма шифрования, и демонстрационного приложения **RijndaelDemo.exe**. Для работы приложения необходима ОС Windows с установленным .NET Framework v1.1.

В основном окне демонстрационной программы задаются длина ключа, длина блока, а также расширенный ключ шифрования, вычисляемый в соответствии с заданным ключом шифрования (рис. 1.31, 1.32).

Можно подробно рассмотреть действие всех цикловых преобразований (ByteSub, ShiftRow, MixColumn, AddRoundKey) как при шифровании, так и при расшифровании (рис. 1.33, 1.34).

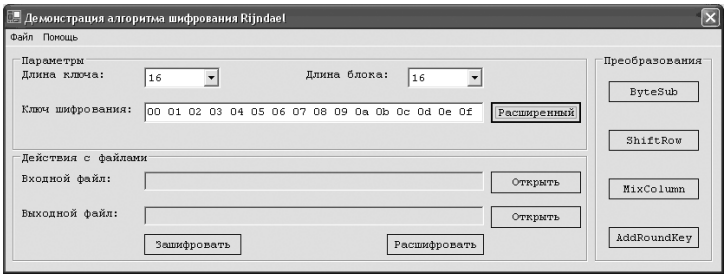


Рис. 1.31. Главное окно демонстрационной программы

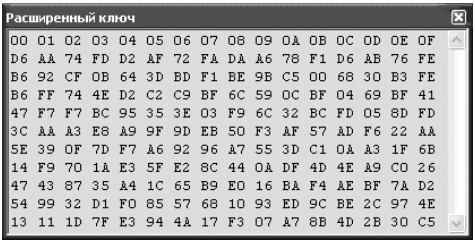


Рис. 1.32. Окно расширенного ключа

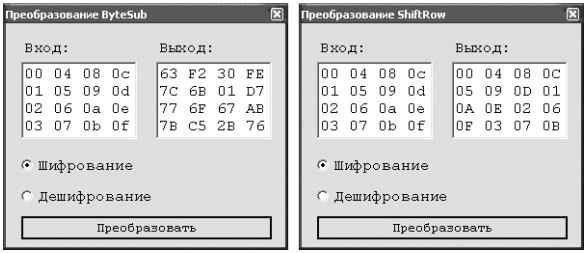


Рис. 1.33. Окна преобразований ByteSub и ShiftRow

При шифровании предлагается выбрать исходный файл и файл, куда будет помещен результат шифрования, при расшифровании — соответственно зашифрованный файл и файл, предназначенный для помещения результата расшифрования. В процессе используются указанные в главном окне программы ключ шифрования и длины ключа и блока.

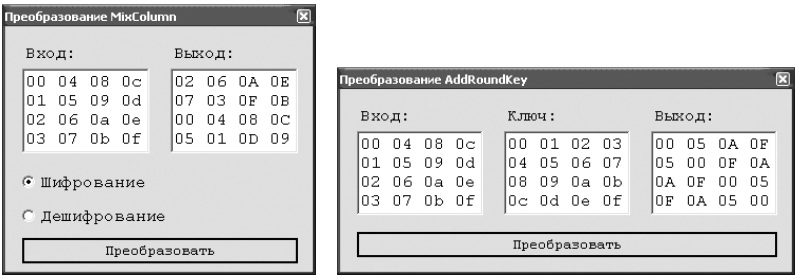


Рис. 1.34. Окна преобразований MixColumn и AddRoundKey

Задание

1. Ознакомиться со сведениями о программе RijndaelDemo. Запустить модуль **RijndaelDemo.exe**.

Примечание. Для обеспечения корректной функциональной работы программы на операционной системе Windows 10 при запуске RijndaelDemo необходимо включить режим совместимости.

2. Изучить на примере обычных текстовых файлов способы шифрования и расшифрования с помощью алгоритма Rijndael. Подробно рассмотреть действие всех цикловых преобразований (ByteSub, ShiftRow, MixColumn, AddRoundKey) как при шифровании, так и расшифровании. Исходный текст для шифрования может быть подготовлен заранее и сохранен в файле *.txt.

3. Сохранить в отчете экранные формы, демонстрирующие процесс шифрования и расшифрования информации, проанализировать полученные результаты.

4. Включить в отчет о лабораторной работе ответы на контрольные задания, выбранные в соответствии с номером варианта, указанным преподавателем (табл. 1.9).

Таблица 1.9

Номер варианта	Контрольные задания
1, 5, 7, 26	Сравнить основные характеристики алгоритмов Rijndael и ГОСТ 28147—89
2, 4, 6	Сравнить основные характеристики алгоритмов Rijndael и DES
11, 13	Описать структуру сети Фейстеля
12, 14, 16	Привести обобщенные схемы шифрования данных с помощью алгоритма Rijndael и ГОСТ 28147—89. Дать их сравнительный анализ

Окончание

Номер варианта	Контрольные задания
3, 9, 18, 29	Сравнить один раунд шифрования данных с помощью алгоритма Rijndael и ГОСТ 28147—89
20, 22, 24	Сравнить эквивалентность прямого и обратного преобразований в алгоритмах Rijndael и ГОСТ 28147—89
10, 17, 19	Сравнить выработку ключевой информации в алгоритмах Rijndael и ГОСТ 28147—89
21, 23, 25	Сравнить алгоримы Rijndael и ГОСТ 28147—89 по показателям диффузии
8, 28, 27	Сравнить алгоримы Rijndael и ГОСТ 28147—89 по показателям стойкости
12, 15, 30	Сравнить алгоримы Rijndael и ГОСТ 28147—89 по показателям производительности и удобству реализации

ЧАСТЬ 2

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Асимметричные системы шифрования. Смысл *асимметричных криптосистем* (*системы открытого шифрования с открытым ключом — public key systems*) состоит в том, что для зашифрования и расшифрования используются разные преобразования. Одно из них — зашифрование — является абсолютно открытым для всех. Другое же — расшифрование — остается секретным за счет секретности ключа расшифрования. Таким образом, любой, кто хочет что-либо зашифровать, пользуется открытым преобразованием, но расшифровать и прочитать это сможет лишь тот, кто владеет секретным ключом. Схема асимметричной криптосистемы представлена на рис. 2.1.

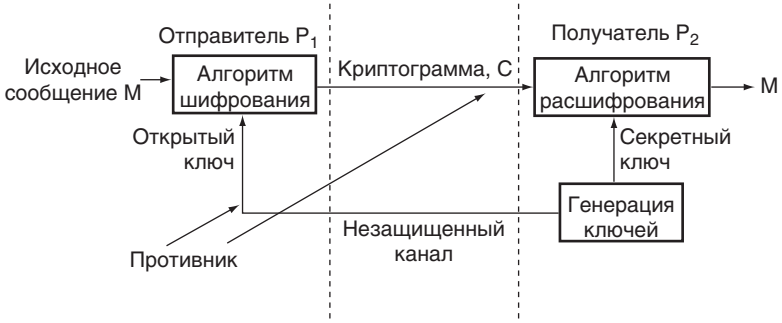


Рис. 2.1. Обобщенная схема асимметричной криптосистемы

В настоящее время во многих асимметричных криптосистемах вид преобразования определяется ключом. У пользователя есть два ключа — секретный и открытый. Открытый ключ публикуется в общедоступном месте, и каждый, кто хочет послать сообщение этому пользователю, зашифровывает текст открытым ключом. Расшифровать сообщение может только упомянутый пользователь с секретным ключом. Таким образом, отпадает проблема передачи секретного ключа, как в симметричных системах. Однако, несмотря на все свои преимущества, эти криптосистемы достаточно трудоемки и медлитель-

ны. Стойкость асимметричных криптосистем базируется в основном на алгоритмической трудности решить за приемлемое время какую-либо задачу. Если злоумышленнику удастся построить такой алгоритм, то дискредитирована будет вся система и все сообщения, зашифрованные с помощью этой системы. В этом состоит главная опасность асимметричных криптосистем в отличие от симметричных.

Алгоритм Диффи — Хеллмана. Алгоритм Диффи — Хеллмана (Diffie — Hellman) использует функцию дискретного возведения в степень. Вначале генерируются два больших простых числа n и q. Эти два числа не обязательно хранить в секрете. Далее один из партнеров P_1 генерирует случайное число x и посылает другому участнику будущих обменов P_2 значение

$$A = q^x \bmod n.$$

По получении значения A партнер P_2 генерирует случайное число y и посылает участнику обмена P_1 вычисленное значение

$$B = q^y \bmod n.$$

Партнер P_1, получив значение B, вычисляет $K_x = B^x \bmod n$, а партнер P_2 — $K_y = A^y \bmod n$. Алгоритм гарантирует, что числа K_y и K_x равны и могут быть использованы в качестве секретного ключа для шифрования. Даже перехватив числа A и B, трудно вычислить K_x или K_y. Схематично работа алгоритма Диффи — Хеллмана представлена на рис. 2.2.

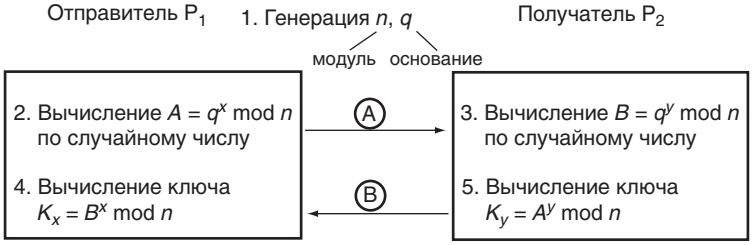


Рис. 2.2. Алгоритм Диффи — Хеллмана

Пример 2.1
$n = 5$, $q = 7$, $x = 3$, $y = 2$
$A = 7^3 \pmod 5 = 343 \pmod 5 = 3$; $B = 7^2 \pmod 5 = 49 \pmod 5 = 4$;
$K_x = 7^3 \pmod 5 = 64 \pmod 5 = 4$; $K_y = 3^2 \pmod 5 = 4$.

Алгоритм RSA. Первое практическое воплощение принцип открытого шифрования получил в системе RSA, разработанной в 1977 г.

в Массачусетском технологическом институте (США) и получившей свое название от первых букв фамилий авторов: Рональд Ривест (R. Rivest), Эди Шамир (A. Shamir), Леонард Адлеман (L. Adleman).

Идея авторов этого алгоритма состояла в том, что, взяв целое число N в виде произведения двух больших простых чисел $N = P \cdot Q$, легко подобрать пару чисел Y и X, таких, чтобы для любого целого числа M, меньшего N, было справедливо соотношение

$$(M^X)^Y = M \bmod N.$$

В качестве открытого ключа шифрования в системе RSA выступают ключ Y и модуль N, а секретным ключом для расшифрования сообщений является число X. Процедура шифрования сообщения M, рассматриваемого как целое число (такое допущение возможно вследствие того, что любой контент может быть представлен в числовой форме при обработке в средствах вычислительной техники), меньшее N (при необходимости длинное сообщение разбивается на отрезки, шифруемые независимо), состоит в вычислении значения

$$C = M^Y \bmod N.$$

Расшифрование осуществляется аналогично с использованием секретного ключа X:

$$M = C^X \bmod N.$$

Математически строго можно доказать, что определение по паре чисел (N, Y) секретного ключа X не проще разложения на простые множители числа N, т.е. нахождения P и Q. Задача же разложения на множители целого числа изучается в математике с древнейших времен и известна как сложная вычислительная задача. В настоящее время разложение числа из нескольких сотен десятичных знаков потребует от современных вычислительных машин сотен лет непрерывной работы.

Далее представлен пример работы алгоритма RSA.

Генерация ключей

Получатель 1. P, Q — простые, $N = P \cdot Q$

2. $\varphi(N) = (P - 1) \cdot (Q - 1)$, $\varphi(N)$ — функция Эйлера

Выбор открытого ключа Y:

$1 < Y \le \varphi(N)$, НОД(Y, $\varphi(N)$) = 1

Выбор открытого ключа X:

$X \cdot Y \equiv 1 \pmod{\varphi(N)}$

Отправитель **шифрование** M ($M_i = 0, 1, 2, \ldots, N - 1$)

3. $C_i = M_i^Y \pmod N$

Получатель **расшифрование** C (C_1, C_2, ..., C_i, ...)
4. $M_i = C_i^X \pmod N$

Пример
Генерация ключей
1. $P = 3$, $Q = 11$, $N = P \cdot Q = 33$
2. $\varphi(N) = (P - 1) \cdot (Q - 1)$, $\varphi(N) = 1$
$Y = 7$, НОД(Y, $\varphi(N)$) = 1
$X \cdot Y = 1 \pmod{20}$, $7 \cdot 3 = 1 \pmod{20}$, X = 3
Сообщение: $M_1 M_2 \rightarrow 32$; $M_1 = 3 < 33$, $M_2 = 2 < 33$
Шифрование $C_i = M_i^Y \pmod N$
3. $C_1 = 3^7 \bmod 33 = 2187 \bmod 33 = 9$
$C_2 = 2^7 \bmod 33 = 128 \bmod 33 = 29$
Расшифрование $M_i = C_i^X \pmod N$
4. $M_1 = 9^3 \bmod 33 = 729 \bmod 33 = 3$
$M_2 = 29^3 \bmod 33 = 24389 \bmod 33 = 2$

Методы проверки чисел на простоту. Одна из главных проблем асимметричного шифрования — генерация больших простых чисел. Простейшим методом проверки простоты натурального числа N является метод пробных делений: для $d = 2, 3 \ldots$ проверяется выполнение условия (d, N) >1 (здесь (d, N) — наибольший общий делитель чисел d, N). Число операций, требуемых для этого метода, имеет порядок $\sqrt{N}$. Поэтому уже для чисел порядка 10^{30}—10^{40} этот метод неприменим.

В отличие от таких детерминированных методов существуют еще вероятностные методы проверки простоты. Для исследуемого числа проверяется выполнение некоторых условий, связанных со случайными числами. Если какое-либо из этих условий не выполнено, то N — составное число. Если же все условия выполнены, то с некоторой вероятностью можно утверждать, что N — простое число. Эта вероятность тем ближе к единице, чем большее количество случайных чисел мы проверим. Обычно эти условия основаны на малой теореме Ферма, утверждающей, что для любого положительного числа b, не превосходящего некоторого простого числа p,

$$b^{(p-1)} = 1 (\bmod p).$$

Например, $2^6 = 64 = 63 + 1 = 1 (\bmod 7)$. Если требуется определить, является ли целое число r простым, то можно выбрать любое положительное целое число b, меньшее r, и проверить, выполнено ли равенство

$$b^{(r-1)} = 1 (\bmod r).$$

Если равенство не выполнено, то на основании теоремы Ферма можно быть совершенно уверенным, что r — не простое число. Если же равенство выполнено, то можно лишь предполагать, что r — простое число, и поэтому назвать его псевдопростым по основанию b. Вероятность $P(x)$ того, что составное число x окажется псевдопростым по случайному основанию, убывает с ростом x.

К сожалению, существуют так называемые числа Кармайкла — составные числа, которые обладают свойством

$b^{(r-1)} = 1 (\mathrm{mod}\ r)$ для всех b из интервала $[1, r]$, которые взаимно просты с r.

Примером числа Кармайкла является число $561 = 3 \cdot 11 \cdot 17$.

Классический результат теории чисел — теорема Чебышева — показывает, что доля положительных целых чисел, меньших некоторого целого m и являющихся простыми, близка к $1/(\ln m)$. Например, доля целых чисел, меньших 10^{100} и являющихся простыми, близка к $1/(\ln 10^{100}) = 1/230$. Таким образом, если выбрать случайно большое целое положительное нечетное число x и последовательно проверять на простоту числа $x, x + 1, x + 2, \ldots$, то в среднем впервые простое число встретится на шаге с номером $\ln x$.

ЛАБОРАТОРНАЯ РАБОТА № 5

ГЕНЕРАЦИЯ ПРОСТЫХ ЧИСЕЛ, ИСПОЛЬЗУЕМЫХ В АСИММЕТРИЧНЫХ СИСТЕМАХ ШИФРОВАНИЯ

Цель работы: изучение методов генерации простых чисел, используемых в системах шифрования с открытым ключом.

Описание лабораторной работы. Для выполнения лабораторной работы необходимо запустить программу **L_PROST.exe**. На экране дисплея появляется окно с текстовым редактором (для отображения информации об осуществленных программой действиях), в верхней строке окна расположено главное меню, чуть ниже основного меню — панель инструментов (для управления быстрыми командными кнопками и другими «горячими» элементами управления), а в самом низу окна, под тестовым редактором, находится строка состояния, в которой указывается подсказка и выводится дополнительная информация. Клавиши панели инструментов для удобства снабжены всплывающими подсказками.

Для того чтобы попасть в основное меню, необходимо нажать клавишу F10. Передвижение по главному меню осуществляется клавишами перемещения курсора. Чтобы вызвать пункт меню, нужно нажать клавишу ENTER, ESC — выход из основного меню.

Для удобства в программе предусмотрена работа с мышью. В этом случае указатель подводится к нужному пункту главного меню или к нужной кнопке на панели инструментов и нажимается левая клавиша мыши, для отказа достаточно нажать клавишу ESC.

Кроме основных функций главного меню (Генерация простого *P*, Поиск в интервале, Проверка на простоту и т.д.) панель инструментов содержит клавишу, при нажатии которой выводится информация о программе.

Генерация простого P. Возможность генерации простого числа; количество разрядов генерируемого числа задается пользователем (от 1 до 5).

Поиск в интервале. Возможность поиска простых чисел в заданном интервале. Пользователем задается начало интервала — значение x, длина интервала — значение L. Поиск будет осуществляться в интервале $(x, x + L)$.

При проверке на простоту каждого числа интервала сначала выполняется тест пробных делений на первые по порядку простые числа, а затем проверка по тесту Ферма. Для задания способов проверки на простоту необходимо левой клавишей мыши отметить флажок

рядом с названием нужного метода, а затем указать все необходимые данные для начала поиска.

В методе пробных делений исходными данными является количество первых простых чисел для деления, а в тесте Ферма надо указать количество оснований и их значения.

По окончании расчета на экран выводятся найденные простые числа и их количество. Полную картину результатов работы можно просмотреть в пункте меню Вывод результатов.

Проверка на простоту. Возможность проверки на простоту любого числа. Необходимо ввести число и параметры расчета аналогично поиску в интервале.

Вывод результатов. Возможность просмотра всех результатов последней обработки. При входе в программу, когда никаких расчетов еще не производилось, предоставляется возможность просмотра исходного файла первых простых чисел, используемых для теста пробных делений.

Дополнительные сведения о программе

1. При запуске утилит генерации простого числа, поиска в интервале и проверки на простоту у пользователя запрашивается подтверждение на правильность выбранного метода для работы.
2. Во время работы длительных по исполнению процедур запускается прогресс процесса и гасится окно текстового редактора. По полоске прогресса можно наблюдать и оценивать примерную скорость работы алгоритма и время окончания текущего процесса.
3. Будьте внимательны при установке параметров работы, так как в процессе вычисления по ходу работы эти параметры изменить уже не удастся.
4. Описание «горячих» клавиш:
   Ctrl+F1 — генерация простого P;
   Ctrl+F2 — поиск в интервале;
   Ctrl+F3 — проверка на простоту;
   Ctrl+F4 — вывод результатов;
   Ctrl+X — выход из программы.
5. В лабораторной работе из-за большого времени счета рекомендуется использовать числа не более пяти разрядов и длину интервала выбирать не более 500, количество оснований для теста Ферма — не более 5.
6. Для правильного функционирования программы в рабочей директории (вместе с файлом **l_prost.exe**) обязательно должны на-

ходиться файлы **prost.txt** и **work.txt**. Не рекомендуется вносить какие-либо изменения в эти текстовые файлы, иначе последствия могут быть непредсказуемые.

Задание

1. Проверить на простоту два произвольных целых числа разрядностью не менее 5.
2. Распределение простых чисел.
   2.1. Задан интервал вида $[x, x + L]$. Вычислить количество $\Pi(x, L)$ простых чисел в интервале и сравнить с величиной $L/\ln(x)$. При каких условиях $\Pi(x, L)/L$ близко к $1/\ln(x)$ при заданных $x = 2000$, $L = 500$, количество простых чисел для деления 5—15, количество оснований 1—2?
   2.2. Найти в интервале (1000, 1000 + 300) все простые числа. Пусть $L(i)$ — разность между двумя соседними простыми числами. Построить гистограмму для $L(i)$. Вычислить выборочное среднее $L_{\text{сред.}}$. Сравнить с величиной $\ln(x)$, где x — середина интервала. Задано: количество простых чисел для деления 5—20, количество оснований 1—3.
   2.3. Для заданного набора чисел $\{k\}$ оценить относительную погрешность формулы для k-го простого числа:
   $p(k) = k/\ln(k)$, $k = \{10, 15, 20, 30, 35\}$.
3. Методы генерации простых чисел.
   3.1. В интервале (500, 500 + 200) построить график относительного количества натуральных чисел, проходящих «решето Эратосфена», т.е. не делящихся на первые k простых.
   Расчет производится для всех $k <= 10$.
   3.2. Для интервала (1500, 1500 + 300):
   а) рассчитать точное количество P_0 простых чисел в интервале, т.е. при проверке задать только тест на делимость. Количество первых простых чисел для деления определяется из расчета максимальное число для деления равно квадратному корню из максимального значения интервала;
   б) составить тест с небольшим количеством пробных делений и одним основанием в тесте Ферма. Вычислить количество P_1 вероятно простых чисел, удовлетворяющих этому тесту;
   в) составить тест с бóльшим, чем в предыдущем случае, количеством пробных делений и двумя или тремя основания-

ми в тесте Ферма. Вычислить количество P_2 вероятно простых чисел, удовлетворяющих этому тесту.
Проанализировать полученные данные.

3.3. Известно, что в заданном интервале имеются числа Кармайкла. Найти их.
Варианты интервалов: (1050, 1050 + 100);
(1700, 1700 + 100);
(2400, 2400 + 100).

4. Привести в отчете ответы на контрольные вопросы в соответствии с номером варианта (табл. 2.1).

Таблица 2.1

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Почему в качестве первого основания в тестах типа теста Ферма для проверки на простоту очень больших чисел целесообразно использовать число 2?
2, 4, 6, 8, 20, 22, 24, 26, 30	Какова вероятность $P(x)$ того, что наугад взятое нечетное очень большое число, не превосходящее x, окажется простым?
11, 13, 15, 10, 17, 19, 27	Вычислить: 1812 (mod 13), 127 (mod 7)
12, 14, 16, 21, 23, 25, 29	Сформулируйте суть теста на простоту с использованием пробных делений

ЛАБОРАТОРНАЯ РАБОТА № 6

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Цель работы: ознакомление с принципами защищенного электронного документооборота в телекоммуникационных сетях и алгоритмами постановки электронной подписи.

Описание лабораторной работы. Обмен электронными документами по телекоммуникационным сетям существенно снижает затраты на обработку и хранение документов, ускоряется их поиск, однако при этом возникает проблема аутентификации, т.е. установления подлинности автора и отсутствия изменений в полученном документе.

При обработке документов в электронной форме непригодны традиционные способы установления подлинности по рукописной подписи и оттиску печати на бумажном документе, здесь принципиально новым решением является электронная подпись.

Электронная подпись — информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию[1].

Первая схема электронной цифровой подписи (ЭЦП)[2] — RSA была разработана еще в конце 1970-х годов, однако проблема подтверждения авторства стала актуальной настолько, что в 1990-х потребовалось установление стандарта. Причиной послужило повсеместное расширение глобальной сети Интернет и массовое распространение электронной торговли и оказания услуг. Именно по указанной причине стандарты ЭЦП в России и США были приняты практически одновременно, в 1994 г.

Из предложенных криптологами схем ЭЦП наиболее удачными оказались RSA и схема Эль-Гамаля. Первая из них была запатентована в США и в ряде других стран (патент на RSA прекратил свое действие совсем недавно). У второй схемы существует большое количество возможных модификаций, и все их запатентовать весьма затруднительно. Именно по этой причине схема ЭЦП Эль-Гамаля осталась по большей части свободной от патентов. Кроме того, эта схема имеет и определенные практические преимущества: размер блоков, которыми оперируют алгоритмы, и соответственно размер ЭЦП в ней

---

[1] Федеральный закон «Об электронной подписи» от 06.04.2011 № 63-ФЗ, принятый взамен ранее действовавшего ФЗ от 10.01.2002 «Об электронной цифровой подписи».

[2] Согласно упомянутому ФЗ № 63, понятия «электронная подпись» и «электронная цифровая подпись» считаются синонимами.

оказались значительно меньше, чем в RSA, при той же самой стойкости. Именно поэтому стандарты ЭЦП России и США базируются на схеме Эль-Гамаля.

Принцип построения ЭЦП. Асимметрия ролей отправителя и получателя в схемах ЭЦП требует наличия двух тесно связанных ключей: *секретного*, или ключа подписи, и *открытого*, или ключа проверки подписи.

Любая схема ЭЦП обязана определить три следующих алгоритма:

1) генерации ключевой пары для подписи и ее проверки;
2) постановки подписи;
3) проверки подписи.

Стандарты России и США очень похожи, они различаются лишь некоторыми числовыми параметрами и отдельными деталями выработки ключевой пары, вычисления и проверки подписи. Действительно, оба стандарта являются вариантами одной и той же схемы ЭЦП Эль-Гамаля.

ЭЦП используется для аутентификации текстов, передаваемых по телекоммуникационным каналам. Функционально она аналогична обычной рукописной подписи и обладает основными ее достоинствами:

- удостоверяет, что подписанный текст исходит от лица, поставившего подпись;
- не дает самому этому лицу возможность отказаться от обязательств, связанных с подписанным текстом;
- гарантирует целостность подписанного текста.

ЭЦП представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом, и включает две процедуры:

1) процедуру постановки подписи, в которой используется секретный ключ отправителя сообщения;
2) процедуру проверки подписи, в которой используется открытый ключ отправителя.

Процедура постановки подписи. При формировании ЭЦП отправитель прежде всего вычисляет хэш-функцию $h(M)$ подписываемого текста M. Вычисленное значения хэш-функции $h(M)$ представляет собой один короткий блок информации m, характеризующий весь текст M в целом. Затем значение m шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного текста M.

Процедура проверки подписи. При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию $m = h(M)$ принятого по каналу текста M, после чего при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению m хэш-функции.

Принципиальным моментом в системе ЭЦП является невозможность подделки ЭЦП пользователя без знания его секретного ключа.

Каждая подпись, как правило, содержит следующую информацию:

- дата подписи;
- срок окончания действия ключа данной подписи;
- информация о лице, подписавшем текст;
- идентификатор подписавшего (имя открытого ключа);
- собственно цифровая подпись.

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

Хэш-функция должна удовлетворять целому ряду условий:

- быть чувствительной к всевозможным изменениям в тексте M;
- обладать свойством необратимости, т.е. задача подбора документа M_1, который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;
- вероятность того, что значения хэш-функции двух различных документов совпадут, должна быть ничтожно мала.

Большинство хэш-функций строится на основе однонаправленной функции $f(.)$, которая образует выходное значение длиной n при задании двух входных значений длиной n. Этими входами являются блок исходного текста M_i и хэш-значение H_{i-1} предыдущего блока текста (рис. 2.3).

Рис. 2.3. Схема формирования хэш-функции

Алгоритм цифровой подписи DSA. Алгоритм цифровой подписи DSA (Digital Signature Authorization) предложен в 1991 г. в США и является развитием алгоритма цифровой подписи Эль-Гамаля.

Отправитель и получатель электронного документа используют при вычислении большие целые числа:

G, P — простые числа по *L*-бит каждое (*L* = 512 ... 1024 бит)*;*

q — простое число длиной 160 бит делитель числа (*P* – 1).

Числа *G, P, q* являются открытыми и могут быть общими для всех пользователей сети.

Описание алгоритма

1. Отправитель выбирает случайное целое число *X,* $1 < X < q$. Число *X* является *секретным ключом* отправителя для формирования электронной подписи.
2. Отправитель вычисляет значение

$$Y = G^X \bmod P.$$

Число *Y* является *открытым ключом* для проверки подписи отправителя и передается всем получателям документа.

3. Для того чтобы подписать документ *M*, отправитель хэширует его в целое хэш-значение *m:*

$$m = h(M),\ 1 < m < q.$$

Затем генерирует случайное целое число *K,* $1 < K < q$ и вычисляет число *r*:

$$r = (G^K \bmod P) \bmod q.$$

4. При помощи секретного ключа X отправитель вычисляет число *s*:

$$s = ((m + r \cdot X)/K) \bmod q.$$

Пара чисел *r*, *s* образуют цифровую подпись $S = (r, s)$ под документом *M*.

5. Доставленное получателю сообщение вместе с подписью представляет собой тройку чисел [*M*, *r*, *s*]. Прежде всего получатель проверяет выполнение соотношений:

$$0 < r < q;\ 0 < s < q.$$

6. Далее получатель вычисляет значения:

$$w = 1/s \bmod q;$$

$$m = h(M) \text{ — хэш-значение;}$$

$$u_1 = (m \cdot w) \bmod q;$$

$$u_2 = (r \cdot w) \bmod q.$$

Затем при помощи открытого ключа Y вычисляется значение

$$v = ((G^{u1} \cdot Y^{u2}) \bmod P) \bmod q$$

и проверяется выполнение равенства $v = r$. Если оно выполняется, то подпись признается подлинной, так как можно строго математически доказать, что последнее равенство будет выполняться тогда и только тогда, когда подпись $S = (r, s)$ под документом M получена при помощи именно того секретного ключа X, из которого был получен открытый ключ Y.

Алгоритм цифровой подписи RSA. Рассмотрим подробно процедуры постановки и проверки электронной подписи с использованием алгоритма RSA.

Сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель электронных документов вычисляет два больших простых числа P и Q, затем находит их произведение $N = P \times Q$ и значение функции Эйлера:

$$\varphi(N) = (P - 1) \times (Q - 1).$$

Далее отправитель вычисляет число E из условий:

$$E \leqslant \varphi(N), \text{ НОД } (E, \varphi(N)) = 1$$

и число D из условий:

$$D < N, E \times D \equiv 1 \pmod{\varphi(N)}.$$

Пара чисел (E, N) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Число D сохраняется отправителем как секретный ключ для подписывания сообщения. Обобщенная схема формирования и проверки цифровой подписи RSA показана на рис. 2.4.

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M (блок информации, файл, таблица) преобразуют с помощью хэш-функции H (·) в целое число m:

$$m = H(M).$$

Затем вычисляют цифровую подпись S под электронным документом M, используя хэш-значение m и секретный ключ D:

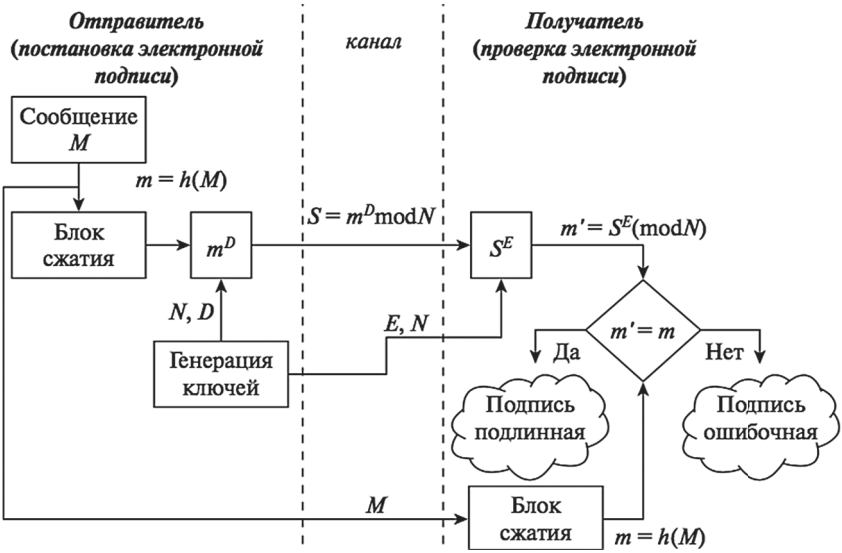


Рис. 2.4. Обобщенная схема формирования и проверки цифровой подписи RSA

$$S = m^D (\text{mod } N).$$

Пара (M, S) передается партнеру-получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа D.

После приема пары (M, S) получатель вычисляет хэш-значение сообщения M двумя разными способами. Прежде всего он восстанавливает хэш-значение m′, применяя криптографическое преобразование подписи S с использованием открытого ключа E:

$$m' = S^E (\text{mod } N) .$$

Кроме того, он находит результат хэширования принятого сообщения M с помощью такой же хэш-функции $H(\cdot) : m = H(M)$.

Задание

1. Ознакомиться с основными направлениями работ в рамках федеральной целевой программы «Электронная Россия», а также со сведениями о порядке использования и действующих алгоритмах постановки электронной цифровой, изложенными выше.

Запустить программу **labWork6.exe**, предназначенную для демонстрации порядка постановки и проверки электронной цифровой подписи.

2. Сгенерировать и переслать участникам обмена ключи для шифрования исходного документа и ключи для подписания документа. Исходный текст для шифрования набирается непосредственно в окне программы.
3. Зашифровать исходное сообщение и подписать его на секретном ключе отправителя.
4. Переслать зашифрованное и подписанное сообщение получателю. Выполнить проверку правильности ЭЦП и восстановить исходный текст сообщения.
5. Сохранить в отчете экранные формы, демонстрирующие процесс генерации и распространения ключей; процесс шифрования исходного документа и постановки ЭЦП.
6. Привести в отчете ответы на контрольные вопросы в соответствии с номером варианта (табл. 2.2).

Таблица 2.2

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	В чем состоит назначение хэш-функций и какие требования предъявляются к хэш-функциям, используемым для постановки ЭЦП? Перечислите стандарты хэш-функций, действующие в Российской Федерации
2, 4, 6, 8, 20, 22, 24, 26, 30	Опишите процедуры постановки и проверки ЭЦП. Какая информация содержится в ЭЦП?
11, 13, 15, 10, 17, 19, 27	Перечислите стандарты ЭЦП, действующие в Российской Федерации. На каких принципах основана криптостойкость современных алгоритмов ЭЦП?
12, 14, 16 21, 23, 25, 29	Приведите пример реализации алгоритма ЭЦП (RSA, Эль-Гамаль, DSA)

ЛАБОРАТОРНАЯ РАБОТА № 7

ШИФРОВАНИЕ МЕТОДОМ СКОЛЬЗЯЩЕЙ ПЕРЕСТАНОВКИ

Цель работы: исследование шифра скользящей перестановки с использованием программной реализации XY-Mover.

Описание лабораторной работы. ***Устойчивые закономерности открытого текста и их использование при дешифровании шифров простой замены и перестановки.*** Возможность дешифрования какого-либо шифра в значительной мере зависит от того, в какой степени криптографические преобразования разрушают вероятностно-статистические закономерности, присутствующие в открытом содержательном тексте. Так, в осмысленных текстах любого естественного языка различные буквы встречаются с разной частотой, при этом относительные частоты букв в различных текстах одного языка близки между собой. То же самое можно сказать и о частотах пар, троек букв открытого текста. Кроме того, любой естественный язык обладает так называемой избыточностью, что позволяет с большой вероятностью «угадывать» смысл сообщения, даже если часть букв в сообщении не известна.

Таблицы относительных частот появления букв в тексте (табл. 2.3) приводятся в разных книгах. Они получены на основе подсчетов частот на больших объемах открытого текста. Учитывая, что для экспериментов берется различный исходный материал, значения вероятностей несколько отличаются между собой.

Таблица 2.3

1	а - 0,062	12	л - 0,035	23	ц - 0,004
2	б - 0,014	13	м - 0,026	24	ч - 0,012
3	в - 0,038	14	н - 0,053	25	ш - 0,006
4	г - 0,013	15	о - 0,090	26	щ - 0,003
5	д - 0,025	16	п - 0,023	27	ы - 0,016
6	е,ё - 0,072	17	р - 0,040	28	ъ,ь - 0,014
7	ж - 0,077	18	с - 0,045	29	э - 0,003
8	з - 0,016	19	т - 0,053	30	ю - 0,006
9	и - 0,062	20	у - 0,021	31	я - 0,018
10	й - 0,010	21	ф - 0,002	32	- 0,175
11	к - 0,028	22	х - 0,009		

Если упорядочить буквы по убыванию вероятностей, то мы получим следующий вариационный ряд:

О,Е,А,И,Н,Т,С,Р,В,Л,К,М,Д,П,У,Я,З,Ы,Б,Ь,Г,Ч,Й,Х,Ж,Ю,Ш,Ц,Щ,Э,Ф

Например, в слове **СЕНОВАЛИТР** содержатся 10 наиболее часто встречающихся букв.

Частоты знаков алфавита зависят не только от языка, но и от характера текста. Так, в тексте по криптографии будет повышена вероятность букв «Ф», «Ш» из-за часто встречающихся слов «шифр», «криптография». В некоторых математических текстах может быть завышена частота буквы «Ф» из-за слов «функция», «функционал» и т.п. В стандартных текстовых файлах наиболее частым является символ «пробел». Частотная диаграмма содержательных текстов является устойчивой характеристикой текста. Из теории вероятностей следует, что при достаточно слабых ограничениях на вероятностные свойства случайного процесса справедлив закон больших чисел, т.е. относительные частоты $\frac{\vartheta_k}{N}$ знаков сходятся по вероятности к значениям их вероятностей p_k

$$P\left\{\left|\frac{\vartheta_k}{N} - p_k\right| > \varepsilon\right\} \xrightarrow[N\to\infty]{} 0.$$

Шифры перестановки и простой замены не полностью разрушают вероятностно-статистические свойства, имеющиеся в открытом сообщении.

При дешифровании текста, зашифрованного шифром простой замены, используют частотные характеристики открытого текста. Именно если подсчитать частоты встречаемости знаков в шифрованном тексте, упорядочить их по убыванию и сравнить с вариационным рядом вероятностей открытого текста, то эти две последовательности будут близки. Скорее всего на первом месте окажется пробел, далее будут следовать буквы «О», «Е», «А», «И».

Конечно, если текст не очень длинный, то не обязательно полное совпадение. Может оказаться на втором месте «О», а на третьем «Е», но в любом случае в первых и вторых рядах одинаковые буквы будут располагаться недалеко друг от друга и чем ближе к началу (чем больше вероятность знаков), тем меньше будет расстояние между знаками.

Аналогичная картина наблюдается и для пар соседних букв, биграмм, открытого текста (наиболее частая биграмма русского открытого текста — СТ). Однако для получения устойчивой картины длина анализируемой последовательности должна быть достаточно большой. На сравнительно небольших отрезках открытого текста эта картина как-то смазана. Более устойчивой характеристикой биграмм является отсутствие в осмысленном тексте некоторых биграмм, как говорят, наличие запретных биграмм, имеющих вероятность, равную практически нулю.

Видели ли вы когда-нибудь в открытом тексте биграмму «ЪЬ» или биграммы вида «гласная» Ь, «пробел» Ь? Знание и использование указанных особенностей открытого текста значительно облегчает дешифрование шифра перестановки и замены.

Шифр перестановки. Положим X — множество открытых (содержательных) текстов в алфавите I. Длины всех возможных открытых текстов кратны величине T. Множеством ключей является симметрическая группа подстановок S_T степени T, для $g \in S_T$ определим функцию шифрования f_g, положив для $(i_1, i_2, ..., i_T) \in X$

$$f_g(i_1, i_2, ..., i_T) = (i_{g(1)}, i_{g(2)}, ..., i_{g(T)}),$$

доопределим f_g на остальных элементах из X по правилу: текст $x \in X$ делится на отрезки длины T и каждый отрезок длины T шифруется на ключе g по указанному выше закону шифрования. Последовательность, составленная из букв образов зашифрованных слов, является шифрованным текстом, соответствующим открытому тексту x и ключу g. Для шифрования текста длины, не кратной T, его дополняют буквами до длины, кратной T.

Дешифрование шифра перестановки. Шифрованный текст записывается в таблицу с T столбцами. Для восстановления открытого текста шифра перестановки нам необходимо переставить колонки таким образом, чтобы в строках появился осмысленный текст.

Рассмотрим пример дешифрования шифра перестановки восьми столбцов. Пусть шифротекст имеет следующий вид (табл. 2.4).

Таблица 2.4

1	2	3	4	5	6	7	8
п	а	я		в		и	м
о	ч	ш	г		у		е
е	б	ж	л		е		о
м		ч		о	т	о	я
	е	г	е		у	с	щ
	а	к	ь	з	а	т	т
я	р	е		е	п		ь
	ю	з	в	а	н	в	
о	й	а	в	е	ш	л	
	е	е	я	м		п	н
ь	р	р	н	з	е	е	е
з	а	м	а	н		а	к
ч	с	т	а		ь	а	н
о	я	л	м		а	л	

Окончание

1	2	3	4	5	6	7	8
о	ь	ч	х	т	а	т	
в		е	о	а	л	е	п
о	е	р	м	т	ь	е	
д	с	г	ы		о	а	т
е	б	в	н		ы		
	а	у	и	н	з	н	л
	г	и	а	о	к	к	д
	а	о	б	д	г	н	
	ж	а	у	е	д	я	д
х	л	и		е	м	о	а
к	р	т	д		ь	о	е
	ь	х	в	т	о	н	
р	л	е		е	д	а	ю
р		з	е	в		е	д
ш		в	а	е	н	е	н
т	и	й	е	в			д
		в		с	д		

Сопоставим перестановке столбцов таблицу 8×8, при этом поставим на пересечении *i*-й строки и *j*-й столбца единицу, если *j*-я колонка после обратной перестановки должна следовать за *i*-й. Наша задача — восстановить таблицу, отвечающую правильной перестановке столбцов.

Давайте теперь попарно пристраивать один столбец к другому. Если при этом в некоторых строках появляются запретные биграммы, то столбцы не могут в открытом тексте следовать друг за другом и соответствующая клеточка зачеркивается. В нашем примере шестой столбец не может следовать за четвертым, так как иначе в тексте в первой строке будет подряд два пробела. Посмотрим, например, шестую строку. Если бы четвертый столбец следовал за первым, то в тексте были бы слова, начинающиеся с «Ь».

После просмотра всех строк получим табл. 2.5.

Таблица 2.5

	1	2	3	4	5	6	7	8
1	x	x		x	x	x		x
2		x		x		x		
3			x					x
4	x	x		x		x	x	x

Окончание

	1	2	3	4	5	6	7	8
5	x				x	x	x	x
6	x	x		x		x	x	
7	x			x	x	x	x	x
8	x	x			x	x	x	x

Если бы текст был подлиннее и строк было бы побольше, то в каждой строке и в каждом столбце осталось бы ровно по одной незачеркнутой клеточке и перестановка была бы восстановлена.

В таблице 2.5 только можно утверждать, что шестой столбец следует за третьим (обозначим это событие следующим образом: $3 \to 6$), если шестой столбец не является последним. Для шестого столбца может быть два варианта продолжения

$$\begin{array}{c} 8 \\ \uparrow \\ 3 \to 6 \to 5 \end{array}$$

Нам надо рассмотреть оба и постараться отсеять ложный вариант. Если отсеять ложный вариант не удастся, то надо продолжать оба варианта

$$\begin{array}{ccccccc} & & 8 & \to 4 & & 1 & \\ & & \uparrow & & & \uparrow & \\ 3 \to & & 6 & \to 5 & \to 2 & & \to 7 \end{array}$$

В итоге получаем некоторое дерево возможного следования столбцов в открытом тексте.

$$\begin{array}{ccccccccccccc} & & & & & & & & 7 & & & & \\ & & & & & & & & \uparrow & & & & \\ 3 & \to & 6 & \to & 5 & \to & 2 & \to & 8 & \to & 4 & \to & 1 \to 7 \\ & & \downarrow & & \downarrow & & \downarrow & & & & & & \\ & & 8 & & 4 & & 1 \to 7 & & & & & & \\ & & \downarrow & & \downarrow & & & & & & & & \\ & & 4 & & 1 \to & & 7 & & & & & & \\ & & \downarrow & & & & & & & & & & \\ & & 5 & & & & & & & & & & \\ & & \downarrow & & & & & & & & & & \\ & & 2 \to 7 & & & & & & & & & & \\ & & \downarrow & & & & & & & & & & \\ & & 1 \to 7 & & & & & & & & & & \end{array}$$

Каждой ветви дерева соответствует некоторая перестановка столбцов.

Далее проверяем каждый вариант на осмысленность и получаем правильный вариант

$$3 \to 6 \to 5 \to 2 \to 8 \to 4 \to 1 \to 7$$

Заметим, что не обязательно было строить дерево до конца. Например, ветвь $3 \to 6 \to 8 \to 4 \to 5$ можно было отсеять сразу. Разве можно признать осмысленным фрагмент текста, приведенный в табл. 2.6?

Таблица 2.6

3	6	8	4	5
я		м		в
ш	у	е	г	
ж	е	о	л	
ч	т	я		о
г	у	щ	е	з
к	а	т	ь	е
е	а	т		а

Такая процедура отсечения ветвей была бы просто необходима, если бы строк было поменьше и дерево было бы соответственно гораздо ветвистее. Предложенную процедуру легко автоматизировать и сделать пригодной для реализации на ЭВМ. Алгоритм дешифрования должен состоять из следующих этапов.

1. Предварительная работа. Анализируя достаточно представительный объем открытых текстов, построить множество запретных биграмм.

2. Предварительная работа. Составить словарь всех возможных v-грамм для $v = 2, 3, ..., d$, которые могут встретиться в открытом тексте. Число d выбирается исходя из возможностей вычислительной техники.

Построить таблицу 8×8. При этом перебираются последовательно все запретные биграммы и для каждой опробуемой биграммы — последовательно все строки. Если хотя бы в одной строке первый символ биграммы встречается в i-м столбце, а второй — в j-м, то клеточка $i \times j$ таблицы зачеркивается.

3. Выбрать некоторый столбец в качестве начального.

4. Начать процедуру построения дерева путем пристраивания к исходному столбцу всех вариантов столбцов.

5. Для каждого полученного варианта добавить еще один из оставшихся столбцов. Если хотя бы в одной из строк таблицы встретится 3-грамма, которая отсутствует в словаре размещенных 3-грамм, то вариант отсеивается.

6. Для каждого из неотсеянных вариантов добавляем еще один столбец и проводим отсев ложных вариантов по словарю разрешенных 4-грамм.

Если словарь был построен только для $d \leq 3$, то отсев проводится путем проверки на допустимость 3-грамм, встретившихся в последних трех столбцах каждой строки. Продолжаем этот процесс до получения полной перестановки.

В таблице 2.7 приведен восстановленный для нашего примера текст.

Таблица 2.7

	1	2	3	4	5	6	7	8
1	я		в	а	м		п	и
2	ш	у		ч	е	г	о	
3	ж	е		б	о	л	е	
4	ч	т	о		я		м	о
5	г	у		е	щ	е		с
6	к	а	з	а	т	ь		т
7	е	п	е	р	ь		я	
8	з	н	а	ю		в		в
9	а	ш	е	й		в	о	л
10	е		м	е	н	я		п
11	р	е	з	р	е	н	ь	е
12	м		н	а	к	а	з	а
13	т	ь		с	н	а	ч	а
14	л	а		я		м	о	л
15	ч	а	т	ь		х	о	т
16	е	л	а		п	о	в	е
17	р	ь	т	е		м	о	е
18	г	о		с	т	ы	д	а
19	в	ы		б		н	е	
20	у	з	н	а	л	и		н
21	е	к	о	г	д	а		к
22	о	г	д	а		б		н
23	а	д	е	ж	д	у		я

Окончание

	1	2	3	4	5	6	7	8
24	и	м	е	л	а		х	о
25	т	ь		р	е	д	к	о
26	х	о	т	ь		в		н
27	е	д	е	л	ю		р	а
28	з		в		д	е	р	е
29	в	н	е		н	а	ш	е
30	й		в	и	д	е	т	ь
31	в	а	с					

Дальнейшее развитие шифры перестановки получили осуществлением идеи непрерывной локальной перестановки символов открытого текста под действием управляющей последовательности. Для осуществления перемешивания знаков открытого текста в памяти шифратора запоминаются отдельные знаки текста и проводится задержка их передачи в дискретном времени. Введем параметры n_1 и n_2 так, что $n = n_1 + n_2$. В этих шифрах i-й знак передаваемого сообщения переставляется в шифрованном сообщении на j-е место, где $i - n_1 \le j \le i + n_2$.

Управляющую последовательность временем задержки стараются выбрать так, чтобы время задержки каждого символа было случайной величиной с равномерным распределением, т.е. вероятность каждого фиксированного значения времени задержки должна быть близка к $1/n$.

Шифрующий автомат скользящей перестановки. Рассмотрим схему шифрующего автомата, позволяющего при подходящей управляющей последовательности реализовать произвольный шифр скользящей перестановки (рис. 2.5).

На вход узла формирования задержки (УФЗ) в каждом такте t подается вектор $\vec{y} = \left(y_1^t, \ldots y_n^t\right)$,

$$y_i^t \in \{0,1\},\ i = \overrightarrow{2,n-1},\ y_1^t = y_n^t = 1.$$

Узел формирования задержки является конечным автоматом $\left(F_2^n, F_2^n, \{1, \ldots, n\}, \delta, \lambda\right)$, множеством состояний которого является множество всевозможных двоичных векторов — заполнений $\bar{x}(t) = (x_1^t, \ldots, x_n^t)$ нижнего накопителя. В такте t вырабатывается натуральное число — значение γ_t задержки

$$\gamma_t = \max j : \left\{x_j^t = y_j^t = 1, j = \overline{1,n}\right\}.$$

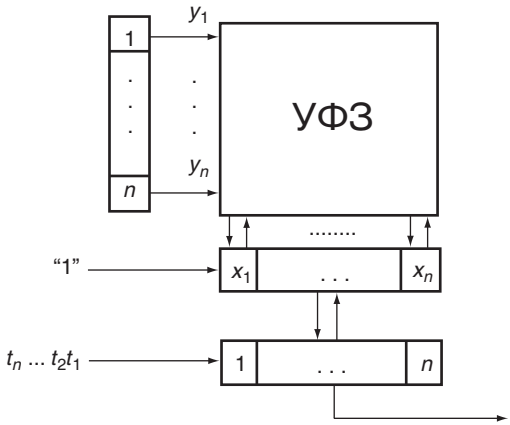


Рис. 2.5. Схема реализации шифрующего автомата скользящей перестановки

При этом автомат переходит в следующее состояние:

$$\bar{x} = (t+1) = (x_1^{t+1}, \ldots, x_n^{t+1}),$$

где

$$x_1^{t+1} = 1,$$

$$x_i^{t+1} = \begin{cases} 0, i = \gamma_t + 1 \\ x_{i-1}^t, i \in \{2, \ldots, \gamma_t, \gamma_t + 2, \ldots, n\}. \end{cases}$$

Знаки открытого текста записываются на нижний накопитель. В линию в t-м такте посылается знак открытого текста, записанный в ячейке с номером γ_t. Состояния автомата $\vec{x}(t)$ являются индикаторами, показывающими, какие из знаков открытого текста еще не считаны с нижнего накопителя.

Для зашифрования последовательности t_N, ..., t_2, t_1 поступают следующим образом. Сначала записываем в нижний накопитель первые n_1 знаков открытого текста

$$(t_{n1}, \ldots, t_1, 0, \ldots, 0).$$

Одновременно автомат устанавливается в начальное состояние

$$\vec{x}(1) = (0, \ldots, 0).$$

После этого автомат УФЗ начинает работать по описанному выше закону до тех пор, пока в накопитель не поступит последний знак t_N

открытого текста. С прекращением подачи на накопитель знаков открытого текста происходит прекращение подачи единиц на накопитель-индикатор. В оставшиеся n_1 тактов производится считывание записанной в накопителе информации.

При расшифровывании УФЗ работает по той же схеме, только вместо считывания необходимо организовывать запись информации во второй накопитель.

Рассмотрим *особенности работы УФЗ*. В каждом такте t (за исключением последних n_1 тактов) в накопителе-индикаторе $\vec{x}(t)$ записано ровно n_1 единиц. Поэтому в такте t величина задержки может принимать только одно из n_1 значений в интервале $\{1, ..., n\}$. В частном случае, когда $n = 1$ либо $n_1 = n$, УФЗ вырабатывает постоянно значения задержки $\gamma_t = 1$ и $\gamma_t = n$ соответственно. Легко видеть, что результирующее преобразование открытого текста действительно будет шифром скользящей перестановки. Условие $y_1^t = 1, t = 1, 2, ...$ обеспечивает постоянное считывание во всех тактах, а условие $y_2^t = 1$ ограничивает величину задержки $\gamma_t \le n$.

Пример 2.2. Примем $n = 7$, $n_1 = 3$, $n_2 = 4$. На вход УФЗ на каждом шаге t работы шифрующего автомата подается вектор $\vec{y} = (y_1^t, ..., y_n^t)$, получаемый в линейном регистре сдвига (ЛРС) (рис. 2.6).

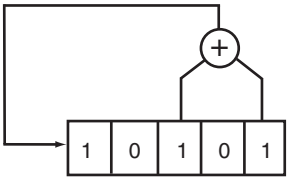


Рис. 2.6. Линейный регистр сдвига

Надеемся, что читатель сможет написать последовательность состояний данного линейного регистра сдвига, с помощью которой образуется управляющая последовательность шифрующего автомата.

Будем обозначать на каждом шаге работы шифрующего автомата последовательность $y_1, y_2, ..., y_n$ (в нашем случае $y_1, y_2, ..., y_7$), поступающую с ЛРС, как «Y», а последовательность единиц $x_1, x_2, ..., x_n$ (в нашем случае $x_1, x_2, ..., x_7$) как «1». В нижней строке будем записывать знаки открытого текста, находящиеся на данном шаге в нижнем накопителе шифрующего автомата $t_1, t_2, ..., t_n$ (в нашем случае $t_1, t_2, ..., t_7$). Рассмотрим пошагово работу шифратора при конкретных условиях.

На каждом шаге, начиная с левого края и идя направо, мы искали первое совпадение в строках Y и 1 (1 в обеих строках) и для удобства обводили этот столбец. Элемент открытого текста, который оказался в выбранном столбце, уходит в линию. Таким образом, в нашем примере последовательность, ушедшая в линию, имеет следующий вид:

$$t_2, t_4, t_5, t_6, t_1, t_7, t_3, t_8.$$

1-й шаг

"Y"	1	1	0	1	0	1	1
"1"	1	1	1	0	0	0	0
	t_3	t_2	t_1	0	0	0	0

2-й шаг

"Y"	1	0	1	0	1	0	1
"1"	1	1	0	1	0	0	0
	t_4	t_3	t_2	t_1	0	0	0

3-й шаг

"Y"	1	0	0	1	0	1	1
"1"	1	0	1	0	1	0	0
	t_5	t_4	t_3	t_2	t_1	0	0

4-й шаг

"Y"	1	0	0	0	1	0	1
"1"	1	0	0	1	0	1	0
	t_6	t_5	t_4	t_3	t_2	t_1	0

5-й шаг

"Y"	1	0	0	0	0	1	1
"1"	1	0	0	0	1	0	1
	t_7	t_6	t_5	t_4	t_3	t_2	t_1

6-й шаг

"Y"	1	1	0	0	0	0	1
"1"	1	1	0	0	0	1	0
	t_8	t_7	t_6	t_5	t_4	t_3	t_2

7-й шаг

"Y"	1	0	1	0	0	0	1
"1"	0	1	0	0	0	0	1
	0	t_8	t_7	t_6	t_5	t_4	t_3

8-й шаг

"Y"	1	0	0	1	0	0	1
"1"	0	0	1	0	0	0	0
	0	0	t_8	t_7	t_6	t_5	t_4

Описание программной реализации. Для выполнения лабораторной работы на компьютере необходимо установить программный модуль XY-Mover. Ниже представлены основные элементы программы.

1. Строка меню. В данной программе меню состоит из трех пунктов: ШИФРОВАНИЕ, ВИД, ПОМОЩЬ. Необходимый пункт меню можно выбрать, щелкнув по нему левой кнопкой мыши, или с помощью кнопок клавиатуры «вправо», «влево», нажав перед этим функциональную клавишу F10. После того как пользователь выбрал необходимый ему пункт меню, откроется ниспадающее подменю. Рассмотрим пункты меню подробнее.

■ ШИФРОВАНИЕ — пункты ниспадающего меню можно выбрать либо левой кнопкой мыши, либо кнопками клавиатуры «↑», «↓». Рассмотрим подробнее пункты подменю:

— РЕДАКТИРОВАНИЕ ПАРАМЕТРОВ — позволяет задать необходимые параметры в поле окна программы «Параметры шифратора», пример настроек представлен на рис. 2.7,

— ШИФРОВАТЬ — запускает процесс шифрования,

— ДЕШИФРОВАНИЕ — запускает процесс дешифрования,

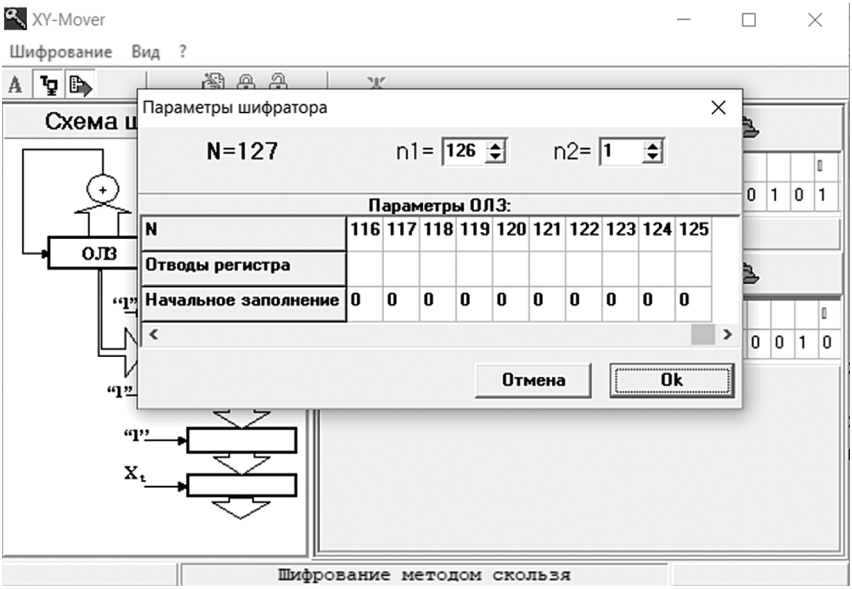


Рис. 2.7. Пример настроек в окне программы «параметры шифратора»

— ВЫХОД — завершение работы программы;

■ ВИД — этот пункт меню позволяет выбрать внешний вид программы. Ниже приводятся пункты подменю:

— ПАРАМЕТРЫ — позволяет использовать поле «Параметры шифратора», описанную ниже;

— СХЕМА ШИФРАТОРА — позволяет наблюдать структурную схему шифрующего автомата;

— СТРОКА СОСТОЯНИЯ — выбор этого пункта меню позволяет наблюдать строку состояния, описанную ниже.

2. Панель инструментов. Возможно, пользователю будет удобнее воспользоваться панелью инструментов (так называемыми кнопками) вместо работы с меню. Кнопки дублируют некоторые пункты меню, но выбрать кнопку гораздо удобнее, чем пункт подменю. Для этого необходимо щелкнуть по выбранной кнопке левой кнопкой мыши. Рассмотрим кнопки слева направо:

■ А, позволяет получить результат, аналогичный пункту меню: ВИД/ПАРАМЕТРЫ;

■ схема шифратора — позволяет получить результат, аналогичный пункту меню ВИД/СХЕМА шифратора;

■ строка состояния — позволяет получить результат, аналогичный пункту меню: ВИД/СТРОКА СОСТОЯНИЯ;

■ редактирование параметров — позволяет получить результат, аналогичный пункту меню ШИФРОВАНИЕ/РЕДАКТИРОВАНИЕ ПАРАМЕТРОВ;

■ шифровать — позволяет получить результат, аналогичный пункту меню ШИФРОВАНИЕ/ШИФРОВАТЬ;

■ «дешифровать» — позволяет получить результат, аналогичный пункту меню ШИФРОВАНИЕ/ДЕШИФРОВАТЬ;

■ «прервать» — позволяет прервать процесс обработки данных (шифрование или дешифрование).

3. Строка состояния. Строка состояния находится в нижней части окна программы. На ней выводится информация о состоянии программы:

■ шифрование методом скользящей перестановки — это сообщение указывает на то, что программа готова к работе.

■ завершено ... % — индикация объема выполненной работы при зашифровании или расшифровании.

4. Описание полей окна программы:

■ параметры шифратора:
  — $n1$ — число знаков, которые записываются в нижний накопитель первыми,
  — $n2$ — остальные знаки (всего их 127),
  — отводы регистра — точки съема ЛРС,
  — начальное заполнение — начальное заполнение ЛРС, которое пользователь может изменять;

■ входной поток:
  — поле, в котором отражается имя текстового файла, содержание которого нужно шифровать (расшифровать),
  — кнопка «Открыть» — при нажатии на эту кнопку открывается стандартное для Windows окно «ОТКРЫТИЕ ФАЙЛА»,
  — текст — содержимое файла, открытого для шифрования (расшифрования),
  — битовый поток — побитное представление символов выбранного файла;

■ выходной поток:
  — поле, в котором отражается имя текстового файла, содержание которого нужно расшифровать/шифровать,
  — кнопка «Открыть» — при нажатии на эту кнопку открывается стандартное для Windows окно «ОТКРЫТИЕ ФАЙЛА»,

— текст — содержимое файла, открытого для расшифрования (шифрования),
— битовый поток — побитное представление символов выбранного файла.

Задание

1. Для выполнения лабораторной работы на компьютере необходимо установить программный модуль XY-Mover.
2. Выполнить начальные установки шифратора согласно примеру.
3. Загрузить файл для шифрования.
4. Произвести шифрование информации с использованием шифра скользящей перестановки, сохранить шифртекст в файле.
5. Описать в отчете процесс шифрования и расшифрования данных с использованием программы-эмулятора XY-Mover. Проанализировать полученные данные.
6. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта (табл. 2.8).

Таблица 2.8

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Почему шифрование методом гаммирования является наиболее подходящим для высокоскоростных линий телекоммуникационной связи?
2, 4, 6, 8, 20, 22, 24, 26, 30	Какие общие требования предъявляются к гамме шифра?
11, 13, 15, 10, 17, 19, 27	Приведите пример, поясняющий работу шифрующего автомата скользящей перестановки при $n = 5$, $n_1 = 2$, $n_2 = 3$
12, 14, 16 21, 23, 25, 29	Кратко опишите работу схемы реализации шифра скользящей перестановки

ЛАБОРАТОРНАЯ РАБОТА № 8

ИЗУЧЕНИЕ ПРОГРАММНЫХ ПРОДУКТОВ ЗАЩИТЫ ИНФОРМАЦИИ. ПРОГРАММА PGP

Цель работы: ознакомление с общими принципами построения и использования программных средств защиты информации, в частности с программой PGPStudio (Pretty Good Privacy) .

Для выполнения лабораторной работы при отсутствии на компьютере программы PGP ее необходимо инсталлировать.

Инсталляционный файл pgpstudio1_2_0_windows.exe.

Освоение средств программной системы PGP, предназначенных:

- для шифрования конфиденциальных ресурсов и разграничения доступа к ним;
- обеспечения целостности информационных ресурсов с помощью механизма электронной цифровой подписи;
- надежного уничтожения остаточной конфиденциальной информации;
- скрытия присутствия в компьютерной системе конфиденциальной информации с помощью виртуального диска.

Описание лабораторной работы. Основные задачи программы PGP — шифровать файлы и почтовые сообщения, заверять их электронными подписями, полностью уничтожать файлы на диске. Программа PGP предоставляет также следующие дополнительные возможности:

- хранение открытых ключей на удаленном сервере;
- использование трех симметричных алгоритмов шифрования и двух асимметричных;
- четыре способа запуска: E-mail plugins, PGPtray, PGPtools, контекстное меню;
- разделение ключей;
- установка уровня валидности ключа и доверия владельцу ключа.

При запуске программы происходит попадание на основную страницу программы, представленную на рис. 2.8.

В верхнем меню содержатся пункт FILE, KEYS, TASKS, HELP.

Раздел FILE содержит функции:

- Preferences (Предпочтения) – при нажатии на пункт меню открывается окно (рис. 2.9). Данное окно позволяет импортировать файлы существующих ключей. Для импорта можно использовать файлы с открытым и закрытым ключом.

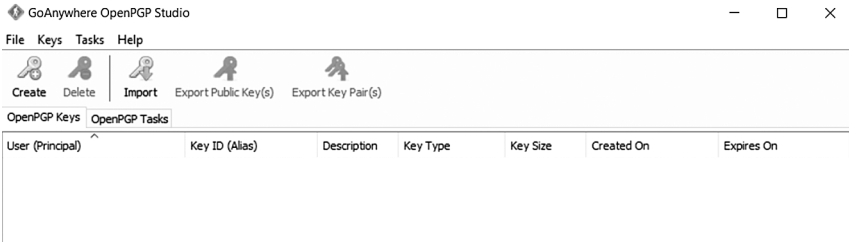


Рис. 2.8. Основное окно работы PGP

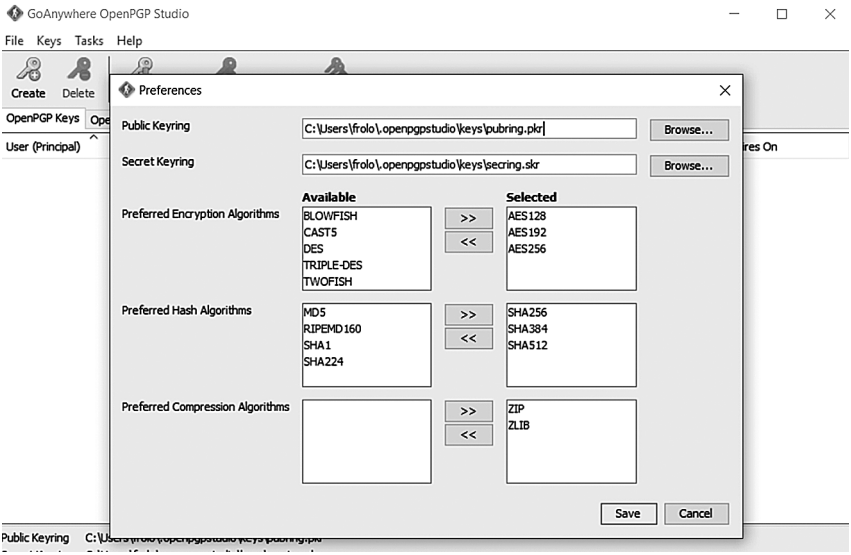


Рис. 2.9. Пункт меню Preferences

Так же есть возможность выбора предпочтительного алгоритма шифрования из списка:

- BLOWFISH (блочное симметричное шифрование с переменной длиной ключа);
- CAST5 (алгоритм симметричного шифрования на основе сети Фейстеля);
- DES (алгоритм симметричного шифрования IBM);
- TRIPLE-DES (симметричный блочный шифр);
- TWOFISH (симметричный алгоритм блочного шифрования с размером блока 128 бит и длиной ключа до 256 бит);

– AES128 (симметричный алгоритм блочного шифрования, размер блока 128 бит),
– AES192 (аналогично AES128, размер блока 192 бит),
– AES256 (аналогично AES128, размер блока 256 бит).

В PGP Studio присутствует возможность выбора предпочтительного алгоритма хеширования из списка: MD5, RIPEMD 160, SHA1, SHA224, SHA256, SHA384, SHA512.

Присутствует возможность выбора алгоритма сжатия из списка: ZIP, ZLIB.

■ EXIT – Выход из программы.
■ Раздел KEYS содержит:
– NEW KEYRING: открывает панель импорта Публичного или Скрытого ключа;
– OPEN KEYRING: позволяет импортировать файл открытого ключа;
– NEW KEY: позволяет создать новый ключ, при нажатии открывается форма, представленная на рис. 2.10;

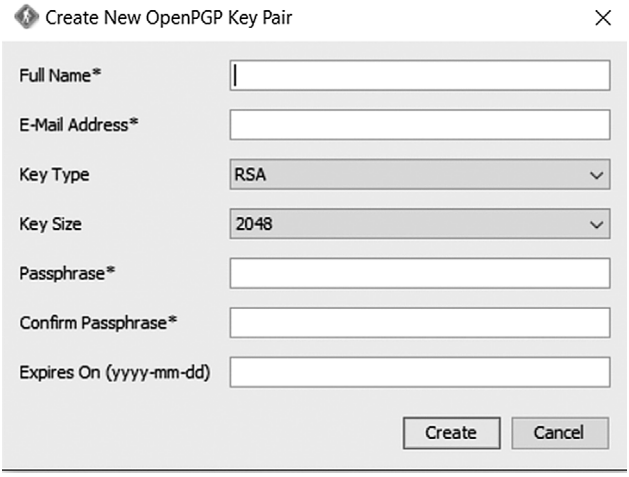


Рис. 2.10. Форма создания нового ключа

– DELITE KEY;
– EXPORT PUBLIC KEY: позволяет экспортировать публичный ключ (по умолчанию не активно);
– EXPORT KEY PAIR: позволяет экспортировать закрытый ключ (по умолчанию не активно);

■ Раздел TASKS (по умолчанию не активно) содержит:
– ENCRYPT (Зашифровать);
– SIGN (Подписать);
– ENCRYPT & SIGN (Зашифровать и подписать);
– DESCRYPT & VERIFY (Расшифровать и проверить);
– VERIFY (Проверить);
■ Раздел HELP

Панель программы содержит две вкладки Open PGP Keys и Open PGP Tasks. Вкладки аналогичны по функционалу соответствующим пунктам меню.

Задание

1. Ознакомиться со сведениями о программе PGP.
2. Запустить программу pgpstudio1_2_0_windows.exe.
3. Создать новый криптографический ключ. Включить в отчет по лабораторной работе сведения о порядке создания ключей шифрования в системе PGP, а также копии используемых при этом экранных форм.
4. Перейти в раздел задач и зашифровать произвольный файл при помощи созданного ключа. Включить в отчет по лабораторной работе сведения о порядке шифрования файлов в системе PGP, а также копии используемых при этом экранных форм.
5. Подписать зашифрованный файл. Включить в отчет сведения о порядке подписания зашифрованных файлов в системе PGP, а также копии используемых при этом экранных форм.
6. Проверить ключ. Включить в отчет сведения о порядке проверки криптографического ключа в системе PGP, а также копии используемых при этом экранных форм.
7. Расшифровать файл при помощи созданного ключа. Включить в отчет сведения о порядке расшифровки файлов в системе PGP, а также копии используемых при этом экранных форм.
8. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта, указанным преподавателем (табл. 2.9).

Таблица 2.9

Номер варианта	Контрольные вопросы
1, 5, 7	Как выбрать длину криптографического ключа в системе PGP?
2, 4, 6, 8	В чем достоинства и недостатки криптографических методов защиты информации?
11, 13, 15, 17, 19	Какие компьютерные системы называются безопасными?
12, 14, 16	В чем заключаются основные требования к защищенности компьютерных систем?
20, 22, 24, 30	Для выполнения каких требований к защищенности компьютерных систем могут применяться криптографические методы защиты?
21, 23, 25	Насколько, на ваш взгляд, надежны методы криптографической защиты информации, используемые в программе PGP?
3, 9, 18, 28	Какими основными функциями защиты информации обладает программа PGP?
10, 26, 27, 29	Какой принцип лежит в основе функции надежного уничтожения остаточной конфиденциальной информации программы PGP?

ЧАСТЬ 3

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В современном мире значение криптографии выходит далеко за рамки обеспечения секретности данных. По мере все большей автоматизации передачи и обработки информации и интенсификации информационных потоков ее методы приобретают уникальное значение. Отметим некоторые современные направления ее приложения:

- защита от несанкционированного чтения, или обеспечение конфиденциальности информации;
- защита от навязывания ложных сообщений как умышленных, так и непреднамеренных;
- идентификация (*identification*) — некое описательное представление какого-либо субъекта;
- контроль целостности информации;
- аутентификация (*authentication*) — проверка подлинности;
- электронная подпись;
- системы тайного электронного голосования;
- электронная жеребьевка;
- защита документов и ценных бумаг от подделки.

Защита от несанкционированного чтения, или обеспечение конфиденциальности информации, обеспечивается путем шифрования информации с использованием современных *симметричных* или *асимметричных* криптографических систем.

Защита от навязывания ложных сообщений может быть обеспечена с помощью так называемой имитозащиты. *Имитозащита* — защита от навязывания ложных сообщений путем формирования в зависимости от секретного ключа специальной дополнительной информации, называемой *имитовставкой*, которая передается вместе с криптограммой, причем могут использоваться два варианта: либо вычисление имитовставки по открытому тексту, либо по шифротексту. Чем больше длина имитовставки, тем меньше вероятность того, что искажение шифротекста не будет обнаружено получателем.

Идентификация законных пользователей заключается в распознавании пользователей, после чего им предоставляются определенные права доступа к ресурсам.

Контроль целостности информации — это обнаружение любых несанкционированных изменений информации, например данных или программ, хранящихся в компьютере. Имитозащита, в сущности, является частным случаем контроля целостности информации, передаваемой в виде шифротекста. В практических приложениях часто требуется удостовериться, что некоторые данные или программы не были изменены каким-либо несанкционированным способом, хотя сами данные не являются секретными и хранятся в открытом виде. Контроль целостности информации может быть основан и на использовании кодов для обнаружения и исправления ошибок.

Аутентификация — установление санкционированным получателем того факта, что полученное сообщение послано санкционированным отправителем. Соблюдение заранее оговоренного *протокола* (набора правил и процедур) должно обеспечить максимальную вероятность этого факта. Очевидно, что при этом контролируется и целостность сообщения на возможность подмены или искажения. Принятый протокол должен обеспечить противодействие использованию потенциальным нарушителем ранее переданных сообщений. Это направление современной криптологии очень интенсивно развивается с момента открытия *криптографии с открытым ключом* (*асимметричной* или *двухключевой криптографии*) в середине 1970-х гг. Если работа Шеннона «Теория связи в секретных системах» (1949) заложила фундамент формирования криптологии как науки, то открытие двухключевой криптографии ознаменовало собой ее переход в качественно новую фазу бурного развития и послужило основой для наиболее полного решения проблем аутентификации информации и разработки систем *электронной цифровой подписи*, которые призваны придать юридическую силу документам и другим сообщениям, переданным электронным способом.

Электронная подпись основывается на двухключевых криптографических алгоритмах, в которых предусматривается использование открытого и секретного ключей.

Системы тайного электронного голосования строятся на базе двухключевых алгоритмов, которые используют механизм слепой подписи, т.е. возможность подписать сообщение без ознакомления с его содержанием. Такие системы имеют большие перспективы для совершенствования системы политического управления современного общества с развитой информационной инфраструктурой.

Электронная жеребьевка сводится, например, к реализации ниже следующего алгоритма.

1. Абонент A выбирает случайное число x_a, двоичное представление которого имеет, например, 80 разрядов, вычисляет значение некоторой односторонней функции $y_a = F(x_a)$ и сообщает величину y_a абоненту B. Абонент B должен угадать, является ли число x_a четным или нечетным.

2. Поскольку используемая функция (известная и B) является односторонней, то B не может по значению y_a определить x_a, поэтому он должен угадывать четность. Пусть абонент B утверждает, что x_a является четным и сообщает об этом абоненту A.

3. Абонент A сообщает абоненту B число x_a.

4. Абонент B вычисляет значение $y = F(x_a)$, если $y = y_a$, то B убеждается, что его партнер действительно предоставил для проверки первоначально выбранное число.

Очевидно, вариантов электронной жеребьевки может быть предложено множество, а практическое использование ее приложимо к любым спортивным жеребьевкам, розыгрышам лотерей и пр.

Криптографическая защита документов и ценных бумаг от подделки является наиболее надежным современным способом пресечения их фальсифицирования. Криптографическая защита от подделки может осуществляться следующим образом. Считывается информация об уникальных особенностях данного носителя информации, формируется цифровой паспорт, включающий содержание документа и информацию о микроструктуре документа. Затем законный изготовитель документа, используя свой секретный ключ, вычисляет цифровую подпись паспорта и записывает на носителе паспорт и соответствующую ему цифровую подпись.

Проверка подлинности документа выполняется путем сканирования микроструктуры материального объекта, на котором сформирован документ, считывания записанной на нем информации и проверки цифровой подписи изготовителя документа по открытому ключу, который является общедоступным. Изготовление фальшивого документа на другом материальном объекте или модифицирование содержания документа и его цифрового паспорта практически неосуществимы без знания секретного ключа, с помощью которого формируется подпись. Любая подделка будет обнаружена путем считывания цифрового паспорта и цифровой подписи, сопоставления паспорта с содержанием документа и проверки подписи по открытому ключу.

Симметричные криптосистемы (*с секретным ключом — secret key systems*) — построены на основе сохранения в тайне ключа шифрования. Процессы шифрования и расшифрования используют один

и тот же ключ. Секретность ключа является постулатом. Основная проблема при применении симметричных криптосистем для связи заключается в сложности передачи обоим сторонам секретного ключа. Однако данные системы обладают высоким быстродействием. Раскрытие ключа злоумышленником грозит раскрытием только той информации, что была зашифрована на этом ключе.

Симметричные криптосистемы в настоящее время принято подразделять на *блочные* и *поточные*.

Блочные криптосистемы разбивают текст сообщения на отдельные блоки и затем осуществляют преобразование этих блоков с использованием ключа.

Блочные шифры оперируют с блоками открытого текста и используют простую замену блоков. Основные процедуры, используемые при получении таких шифров, сводятся к следующему:

рассеивание (*diffusion*) — изменение любого знака открытого текста или ключа влияет на большое число знаков шифротекста, что скрывает статистические свойства открытого текста;

перемешивание (*confusion*) — использование преобразований, затрудняющих получение статистических зависимостей между шифротекстом и открытым текстом.

Поточные криптосистемы работают несколько иначе. На основе ключа системы вырабатывается некая последовательность — так называемая гамма, которая затем накладывается на текст сообщения. Таким образом, преобразование текста осуществляется как бы потоком по мере выработки гаммы.

Практически все современные блочные шифры являются композиционными, т.е. состоят из композиции простых преобразований. Само по себе преобразование может и не обеспечивать нужных свойств, но их цепочка позволяет получить необходимый результат.

ЛАБОРАТОРНАЯ РАБОТА № 9

ШИФР ПЛЕЙФЕРА

Цель работы: изучение принципа шифрования информации с помощью биграммного шифра Плейфера.

Описание лабораторной работы. Шифр Плейфера, или квадрат Плейфера — ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 г. Чарльзом Уитстоном, но названа именем лорда Лайона Плейфера, который внедрил данный шифр в государственные службы Великобритании. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в шифре подстановки и в более сложных системах шифрования Виженера. Таким образом, шифр Плейфера более устойчив к взлому по сравнению с шифром простой замены, так как затрудняется частотный анализ, который может быть проведен, но не для 26 возможных символов (латинский алфавит), а для $26 \times 26 = 676$ возможных биграмм. Анализ частоты биграмм возможен, но является значительно более трудоемким и требует намного большего объема зашифрованного текста.

Шифр Плейфера использует матрицу 5×5 для латинского алфавита (для кириллического алфавита необходимо увеличить размер матрицы до 6×6), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом, составляет матрицу 5×5 и является ключом шифра.

Чтобы зашифровать сообщение необходимо разбить его на биграммы (группы из двух символов), например HELLO WORLD становится HE LL OW OR LD, и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга, затем, руководствуясь ниже сформулированными четырьмя правилами, зашифровываем пары символов исходного текста.

1. Если два символа биграммы совпадают, добавляем после первого символа «X», зашифровываем новую пару символов и продолжаем процесс шифрования. В некоторых вариантах шифра Плейфера вместо вставки «X» используется «Q».

2. Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

3. Если символы биграммы исходного текста встречаются в одном столбце, то они замещаются на символы того же столбца, находящиеся непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

4. Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они замещаются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

Для расшифрования необходимо использовать инверсию этих четырех правил, откидывая символы «X» (или «Q»), если они не несут смысла в исходном сообщении.

Пример работы с программой

Используем **ключ «playfair example»**, тогда матрица примет вид:

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

Зашифруем сообщение «Hide the gold in the tree stump»

HI DE TH EG OL DI NT HE TR EX ES TU MP

1. Биграмма HI формирует прямоугольник, заменяем ее на BM.
2. Биграмма DE расположена в одном столбце, заменяем ее на ND.
3. Биграмма TH формирует прямоугольник, заменяем ее на ZB.
4. Биграмма EG формирует прямоугольник, заменяем ее на XD.
5. Биграмма OL формирует прямоугольник, заменяем ее на KY.
6. Биграмма DI формирует прямоугольник, заменяем ее на BE.
7. Биграмма NT формирует прямоугольник, заменяем ее на JV.

8. Биграмма HE формирует прямоугольник, заменяем ее на DM.
9. Биграмма TR формирует прямоугольник, заменяем ее на UI.
10. Биграмма EX находится в одной строке, заменяем ее на XM.
11. Биграмма ES формирует прямоугольник, заменяем ее на MN.
12. Биграмма TU расположена в одной строке, заменяем ее на UV.
13. Биграмма MP формирует прямоугольник, заменяем ее на IF.

Получаем **зашифрованный текст «BM ND ZB XD KY BE JV DM UI XM MN UV IF»**.

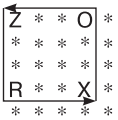
Таким образом, сообщение «Hide the gold in the tree stump» преобразуется в «BMNDZBXDKYBEJVDMUIXMMNUVIF».

Пример работы с программой

Предположим, что необходимо зашифровать биграмму **OR**. Рассмотрим четыре случая:

OR заменяется на **YZ**.

OR заменяется на **BY**.



OR заменяется на **ZX**.

OR заменяется на **ZY**.

Главное окно программы ШИФР ПЛЕЙФЕРА (The Playfair cipher) имеет вид, представленный на рис. 3.1.

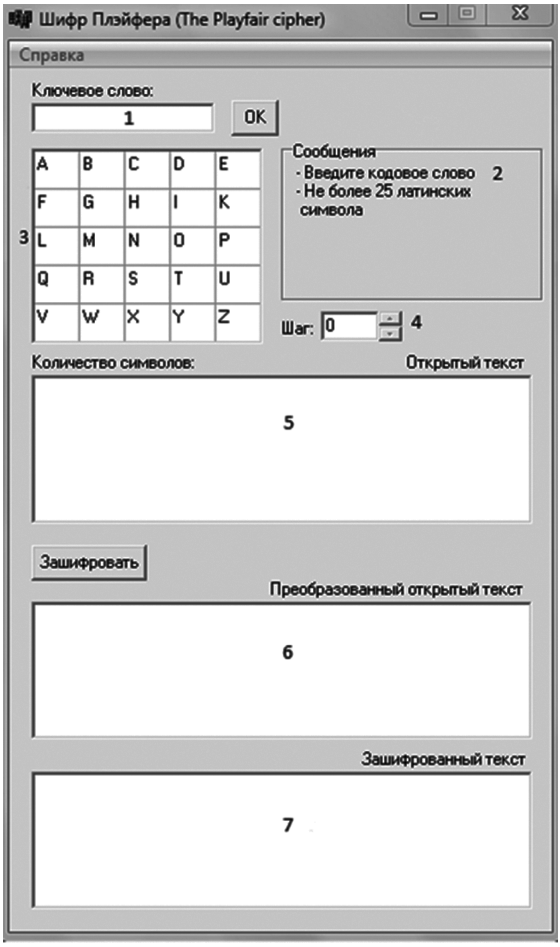


Рис. 3.1. Главное окно программы

Интерфейс программы включает несколько полей.

1. Поле ввода ключевого слова.
2. Форма вывода текстовых сообщений.
3. Ключевая матрица.
4. Кнопка просмотра результатов шифрования в пошаговом режиме.
5. Поле для ввода исходного текста.
6. Поле для вывода преобразованного открытого текста.
7. Поле для вывода зашифрованного текста.

Задание

Для выполнения лабораторной работы на компьютере необходимо установить программу *Playfair. exe*, используемую для демонстрации метода шифрования Плейфера.

1. Для того чтобы начать работу с программой, необходимо ввести ключевое слово в соответствующее поле и нажать кнопку OK. При этом первые ячейки ключевой матрицы займут символы ключевого слова (без повторяющихся символов).

***Внимание*!** Ключевое слово может состоять только из букв латинского алфавита (кроме буквы *J*), длина ключевого слова не более 25 символов.

2. В поле **«Открытый текст»** ввести (или же вставить комбинацией **Ctrl-V**) текст, который необходимо зашифровать. Вводить можно любые символы (буква J, при шифровании, заменится символом I). Нажать на кнопку **«Зашифровать»**.

3. В поле **«Преобразованный открытый текст»** отобразится обработанный текст, а в поле **«Зашифрованный текст»** — результат шифрования.

4. С помощью переключателя **«Шаг»** можно последовательно наблюдать, каким образом шифруется каждая биграмма.

5. Сохранить в отчете экранные формы, демонстрирующие процесс шифрования исходного текста. Сделать выводы по проделанной работе.

6. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 3.1.

Таблица 3.1

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	К какому классу шифров относится шифр Плейфера? Укажите особенности подобных шифров
2, 4, 6, 8, 20, 22, 24, 26, 30	Опишите процедуры шифрования и расшифрования по методу Плейфера
11, 13, 15, 10, 17, 19, 27	Оцените криптостойкость изученного метода шифрования и возможности использования подобных методов в современных криптосистемах
12, 14, 16, 21, 23, 25, 29	Зашифруйте свою фамилию шифром Плейфера вручную. Сравните результаты ручного шифрования и полученные с помощью программы *Playfair.exe*

ЛАБОРАТОРНАЯ РАБОТА № 10

ДЕШИФРОВАНИЕ ШИФРА ПРОСТОЙ ПЕРЕСТАНОВКИ ПРИ ПОМОЩИ МЕТОДА БИГРАММ

Цель работы: знакомство с основами шифрования (расшифрования) данных методом перестановки. Освоение метода дешифрования данных с использованием биграмм.

Описание лабораторной работы

Описание метода шифрования. Перестановочные шифры используются с древних времен. Однако практически в любой современной симметричной криптосистеме можно найти элементы перестановок (например, в алгоритме *DES* — матрицы начальной и конечной перестановок). Приведем пример шифра перестановки. Выберем целое положительное число, скажем, 5(*n*); расположим числа от 1 до 5(*n*) в двухстрочной записи, в которой вторая строка — произвольная перестановка чисел верхней строки:

1 2 3 4 5
3 2 5 1 4

Эта конструкция носит название подстановки, а число 5 называется ее степенью.

Зашифруем фразу «СВЯЩЕННАЯ РИМСКАЯ ИМПЕРИЯ». В этой фразе 23 буквы. Дополним ее двумя произвольными буквами (например, Ь, Э) до ближайшего числа, кратного 5, т.е. до 25. Выпишем эту дополненную фразу без пропусков, разбивая ее на группы по пять символов в соответствии со степенью подстановки:

СВЯЩЕ ННАЯР ИМСКА ЯИМПЕ РИЯЬЭ

Буквы каждой группы переставим в соответствии с указанной двухстрочной записью по следующему правилу: первая буква встает на третье место, вторая — на второе, третья — на пятое, четвертая — на первое и пятая — на четвертое. Полученный текст выписывается без пропусков:

ЩВСЕЯЯННРАКМИАСПИЯЕМЬИРЭЯ

При расшифровании текст разбивается на группы по пять букв и буквы переставляются в обратном порядке: первая на четвертое место, вторая на второе, третья на первое, четвертая на пятое и пятая на третье. Ключом шифра является выбранная степень подстановки

(в нашем случае число 5) и порядок расположения чисел в нижнем ряду двухстрочной записи.

В соответствии с методом математической индукции можно легко убедиться в том, что существует $1 \times 2 \times 3 \times ... \times n(n!)$ вариантов заполнения нижней строки двухстрочной записи. Таким образом, число различных преобразований шифра перестановки, предназначенного для шифрования сообщений длины n, меньше либо равно $n!$ (заметим, что в это число входит и вариант преобразования, оставляющий все символы на своих местах).

Описание метода дешифрования. Сообщения, как бы сложны они ни были, можно представить в виде последовательности знаков. Эти знаки берутся из заранее фиксированного набора, например, кириллического алфавита или палитры цветов (красный, зеленый, синий). Разные знаки встречаются в сообщениях с разной частотой. Поэтому количество информации, передаваемой разными знаками, может быть разным. По известной формуле К. Шеннона количество информации определяется средним числом возможных вопросов с ответами ДА и НЕТ для того, чтобы угадать следующий символ сообщения.

Эффективное кодирование базируется на *основной теореме Шеннона для канала без помех*, суть которой сводится к следующему.

Суть теоремы Шеннона

Сообщения, составленные из букв некоторого алфавита, можно закодировать так, что среднее число двоичных символов на букву сколь угодно близко к энтропии источника этих сообщений, но не меньше этой величины.

Если буквы в тексте следуют независимо друг от друга, то среднее количество информации в сообщении, приходящееся на один знак, равно:

$$H = P_i \operatorname{Log}_2 \frac{1}{P_i},$$

где P_i — частота появления символа i.

Отметим три особенности такого определения количества информации.

1. Оно абсолютно не связано с семантикой, смыслом сообщения, и им можно пользоваться, даже когда точный смысл неясен.

2. В нем предполагается независимость вероятности появления знаков от их предыстории.

3. Заранее известна знаковая система, в которой передается сообщение, т.е. язык, способ кодирования.

В каких единицах выражается значение количества информации по Шеннону? Точнее всего ответ на этот вопрос дает теорема кодирования, утверждающая, что любое сообщение можно закодировать символами 0 и 1 так, что полученная длина сообщения будет сколь угодно близка сверху к *H*. Эта теорема позволяет назвать и единицу информации — *бит*.

Каждый, кто использовал, работая на персональном компьютере, архиваторы, знает, сколь эффективно они сжимают текстовые файлы, ничего при этом не теряя. Их работа лучшим образом демонстрирует теорему Шеннона в действии. Так как для русского текста, переданного лишь прописными буквами, $H = 4{,}43$, и это означает, что, в принципе, в русском алфавите можно было бы обойтись лишь 22 буквами или на 45% сократить длину файлов в кодировке *ASCII*. Таким образом, сообщения занимают места больше, чем это необходимо. Это явление называют избыточностью языка, благодаря чему искажения отдельных символов сообщения зачастую не разрушают содержания, что случилось бы при отсутствии избыточности. Заметьте, у компьютера наиболее часто встречаемые символы *ETOANIRSHDLU* (даны в порядке убывания частот в английском языке) вынесены в центр клавиатуры, чтобы при наборе текстов движение пальцев было бы минимальным. Это расположение клавиш было предложено изобретателем линотипа Оттомаром Мергенталером, который использовал избыточность языка для облегчения работы. Утверждение, что вероятность появления символа в связном тексте не зависит от его предыстории, неверно и статистически, и лингвистически. Уже давно филологи заметили, что обычно за согласной буквой следует гласная, а за гласной согласная. Поэтому в конце XIX в. петербургский математик А.А. Марков предложил рассматривать текст как цепочку символов, где вероятность появления буквы зависит от предыдущей и только от нее. Таким образом, он стал рассматривать не вероятности P_j появления в сообщении знака j, а вероятности P_{ij} появления знака j при условии, что перед ним стоит знак i. Теория марковских цепей оказалась чрезвычайно продуктивной для криптографии, и к отдельным ее применениям мы будем возвращаться позже. Пока же достаточно отметить, что первое свое опробование она имела при анализе текстов «Евгения Онегина» самим Андреем Андреевичем Марковым. Объем информации в одном символе марковской цепи определяется следующей формулой:

$$H = P_i\left(P_{ij} \operatorname{Log}_2 \frac{1}{P_{ij}} \right).$$

В этом случае нет противоречия с требованием независимости знаков, так как знаком здесь считается не отдельный символ, а биграмма. В приложении приведена таблица вероятности встречаемости биграмм в русском техническом тексте. Вероятности представлены десятью классами от 0 до 9 в порядке возрастания и образуют по средним значениям геометрическую прогрессию. Справа в этой таблице даны вероятности встречаемости отдельных символов. Так, из нее следует, что биграмма АЙ встречается довольно часто (класс 7), а биграмма ЙА почти совсем не попадается (класс 0). Среднее количество информации, приходящееся на один символ, определяемое по этой таблице, равно 3,5 бит.

Описанное свойство зависимости буквы в тексте от предыдущей называется марковостью первого порядка, а независимость букв друг от друга — марковостью нулевого порядка. Естественно, что можно рассматривать также и марковости высших порядков, например второго, когда буква зависит от двух предыдущих. Для того чтобы оценить порядок марковости в связном тексте, можно провести случайное моделирование, используя сначала вероятности отдельных букв, потом биграмм, триграмм и т.д. Очевидно, что увеличение порядка марковости повышает схожесть отрывка случайного текста с естественным. Повышение порядка марковости позволяет доуточнить объем информации в сообщениях, но это очень скользкая тема и есть масса разных точек зрения на нее. Действительно, вводя понятие шенноновской меры информации, мы отказались от понятия смысла, который связывает символы в слоги, слоги в слова, слова в предложения, а предложения в сообщение. Практически нет разницы, как сказать ребенку: «Нужно есть кашу!» или «Надо есть кашу!», а вот шенноновский подход эти сообщения считает различными. Поэтому оценка объема информации, содержащейся в сообщении и полученной по приведенным формулам, явно завышена.

Возьмем пример шифра двойной перестановки. Пусть имеется шифровка **АЗЮЖЕ СШГГООИПЕР**, которая так укладывается в таблицу 4×4:

	1	2	3	4
1	А	З	Ю	Ж
2	Е		С	Ш
3	Г	Т	О	О
4	И	П	Е	Р

Рассматривая маловероятные сочетания букв, легко найти истинную последовательность столбцов. Так, сочетание ГТ в третьей строке шифровки указывает на то, что после первого столбца вряд ли следует второй столбец. Рассчитаем статистически, какой столбец скорее всего следует за первым. Для этого воспользуемся таблицей логарифмов вероятностей биграмм русского текста, приведенной в приложении. Вероятность следования одного столбца за другим равна сумме коэффициентов биграмм в строках этих столбцов. Для коэффициентов следования за первым столбцом второй, третий и четвертый имеем выражения:

$$k(1—2) = k(\text{АЗ}) + k(\text{Е}) + k(\text{ГТ}) + k(\text{ИП}) = 7 + 9 + 0 + 5 = 21;$$
$$k(1—3) = k(\text{АЮ}) + k(\text{ЕС}) + k(\text{ГО}) + k(\text{ИЕ}) = 6 + 8 + 8 + 8 = 30;$$
$$k(1—4) = k(\text{АЖ}) + k(\text{ЕШ}) + k(\text{ГО}) + k(\text{ЯР}) = 7 + 5 + 8 + 7 = 27.$$

В нашем случае наиболее вероятно, что после столбца 1 следует столбец 3. Для такой небольшой таблицы шифрования, которую имеем, можно перебрать все варианты перестановок — их всего лишь 24. В случае большого числа столбцов целесообразно оценить вероятности пар сочетаний разных столбцов и решить оптимизационную задачу, которая укажет перестановку столбцов, дающую фрагменты естественного текста с большей вероятностью. В нашем случае наилучший результат достигается при расстановке столбцов (2 4 1 3), что примерно вдвое по вероятностной оценке достовернее ближайшей к ней по вероятности расстановки (4 1 3 2). После того как столбцы шифровки расставлены, не составит труда правильно расставить и ее строки по смыслу фрагментов текста:

	2	4	1	3
1	З	Ж	А	Ю
2		Ш	Е	С
3	Т	О	Г	О
4	П	Р	И	Е

Текст в ней уже читается и, расставив строки в порядке (4 1 2 3), получим расшифровку **ПРИЕЗЖАЮ ШЕСТОГО**.

Можно привести еще один пример расшифровки с использованием биграмм. Зашифруем текст методом перестановки:

ТЕОРИЯ ИНФОРМАЦИИ

Дополним текст до полной матрицы при помощи некоторых случайных букв, в нашем случае «ПОР».

Т	Е	О	Р
И	Я		И
Н	Ф	О	Р
М	А	Ц	И
И	П	О	Р

Ключ: 1 2 3 4
4 2 1 3

Получим:

Зашифрованный текст: ОЕРТ ЯИИОФРНЦАИМОПРИ

О	Е	Р	Т
	Я	И	И
О	Ф	Р	Н
Ц	А	И	М
О	П	Р	И

Для упрощения задачи предположим, что длина ключа известна. Далее, как и в предыдущем примере, необходимо оценить вероятности биграмм в столбцах и решить оптимизационную задачу.

Например, для первого столбца:

$$k(1—2) = k(\text{ОЕ}) + k(\text{Я}) + k(\text{ОФ}) + k(\text{ЦА}) + k(\text{ОП}) = \\ = 6 + 0 + 2 + 5 + 4 = 17;$$

$$k(1—3) = k(\text{ОР}) + k(\text{И}) + k(\text{ОР}) + k(\text{ЦИ}) + k(\text{ОР}) = \\ = 8 + 8 + 8 + 7 + 8 = 39;$$

$$k(1—4) = k(\text{ОТ}) + k(\text{И}) + k(\text{ОН}) + k(\text{ЦМ}) + k(\text{ОИ}) = \\ = 8 + 8 + 7 + 0 + 6 = 29.$$

Таким образом, самый большой коэффициент следования у столбцов 1—3, проводя дальнейшие вычисления, несложно будет определить и весь ключ.

Применение данного метода достаточно эффективно при дешифровании осмысленных сообщений, так как при шифровании методом перестановки вероятностная характеристика символов в осмысленном сообщении сохраняется, что существенно упрощает взлом.

Описание демонстрационной программы. Программный модуль ***cns.exe*** используется для автоматизации процесса шифрования (расшифрования) методом перестановки и вскрытия шифротекстов методом биграмм. Главное окно программы и окно дешифрования представлены на рис. 3.2.

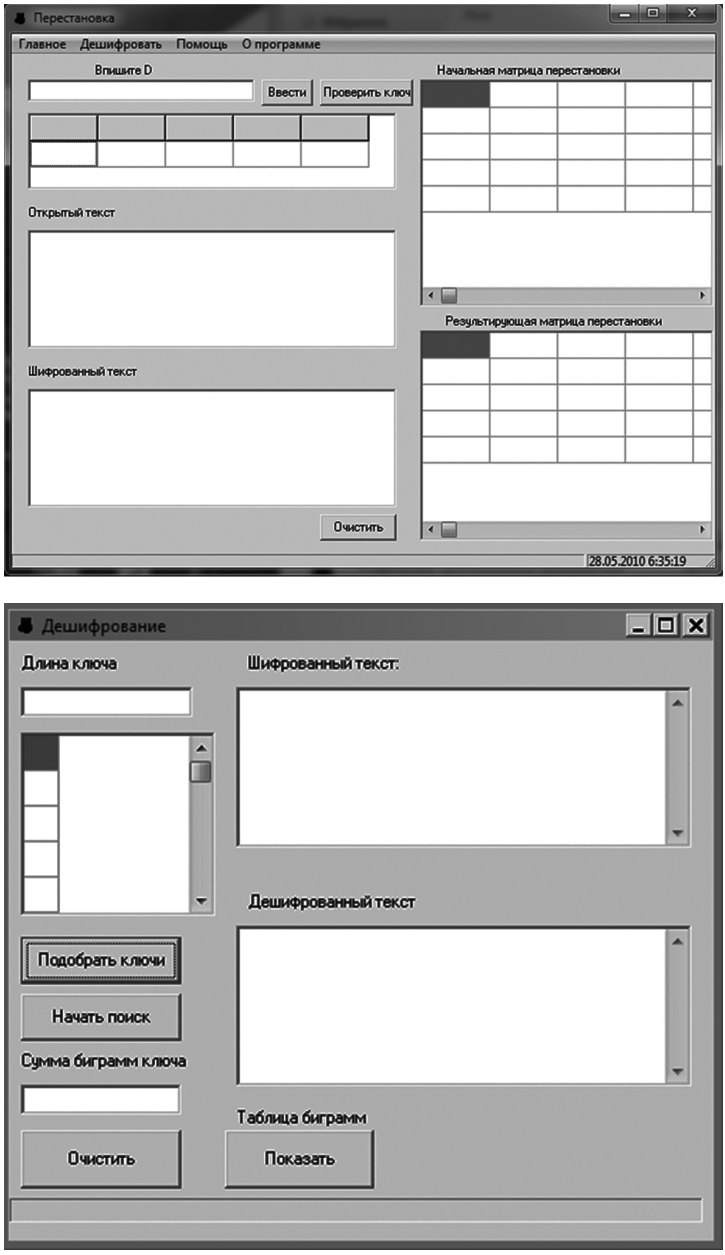


Рис. 3.2. Главное окно программы и окно дешифрования

Дополнительные сведения о программе и авторах программной реализации представлены на рис. 3.3.

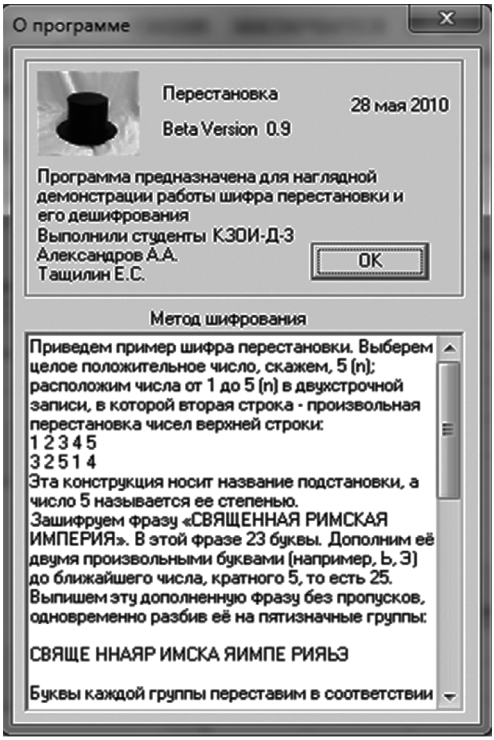


Рис. 3.3. Дополнительные сведения о программе и авторах программной реализации

Задание

Для выполнения лабораторной работы на компьютере необходимо установить файл *cns.exe*.

Запустить программу *cns.exe*, предназначенную для демонстрации шифрования (расшифрования) данных методом перестановки и освоения метода дешифрования данных с использованием биграмм.

Примечание. Для обеспечения корректной функциональной работы программы на операционной системе Windows 10 при запуске cns.exe необходимо включить режим совместимости.

1. Зашифровать (расшифровать) вводимый с клавиатуры текст и тексты, открываемые из файлов.

***Внимание*!** Предварительно просмотрите приложенные к программе тестовые примеры о формате и структуре шифруемых файлов.

Привести в отчете экранные формы выполняемых действий.

2. Произвести попытку дешифрования данных, зашифрованных в п. 1 (*если попытка вскрытия криптограмм не удалась*, *сделать выводы о причинах неудачи*). Привести в отчете экранные формы.

3. Добиться правильного дешифрования зашифрованного текста (*подбором длины и содержания исходного текста*, *длины ключа шифрования*). Приложить экранные формы, сделать выводы об особенностях исследуемого метода вскрытия криптограмм.

4. Сохранить в отчете экранные формы, демонстрирующие процесс шифрования, расшифрования и дешифрования.

5. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 3.2

Таблица 3.2

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	В чем заключается описанный метод вскрытия криптограмм?
2, 4, 6, 8, 20, 22, 24, 26, 30	В чем заключается метод шифрования (расшифрования) с использованием перестановок? Какие перестановочные методы шифрования вы знаете?
11, 13, 15, 10, 17, 19, 27	Приведите примеры использования алгоритма перестановки в современных симметричных криптосистемах
12, 14, 16, 21, 23, 25, 29	Какие требования к исходным текстам и длинам ключей шифрования обеспечат максимальный эффект для использования изученного метода дешифрования?

Приложение 3.1

Таблица коэффициентов встречаемости биграмм в тексте

	АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШТЬЫЬЭЮЯ	Процент
А	27867777477788376782667755000067 9	71
Б	71101622605635727507054105572203 5	13
В	80504803716756846666030130082004 8	38
Г	60116500604544807006001200000000 4	10
Д	81634810704717846527133340064045 7	25
Е	55678666477889658893365656001155 9	73
Ж	60006721705027101213000000002000 2	8
З	84626411615566715006002100262004 6	14
И	66766857777688557881577763010067 9	64
и	00303000003654000660001230000000 8	11
к	81511652712705807667006010000000 7	29
л	84121861804416700336300311068078 6	31
м	75722801704476851316100000073006 8	32
н	90336811906017800576525300085046 7	50
о	28888677687887678883256765001525 9	89
п	70000804703614849456201000045304 4	28
р	91644860805266842667354242074016 7	48
с	64625720707866875696351550056138 7	43
т	82714800806456938846000402178015 8	60
у	34466765336556067771550636000074 8	19
ф	60000500600220604035400000010000 2	4
х	43300400301105605313002000100000 8	6
ц	50600600700000300004000000050000 5	6
ч	70100800706106201073000130013000 4	12
ш	50000600703303403034000000004000 5	3
щ	60000700600002002004000000004010 1	3
ъ	00000400000000000000000000000003 2	1
ы	14735715175562155560070541010001 8	18
ь	01000307106047100640000161000062 8	12
э	00400100026521020170430000000000 1	4
ю	05002012041000000370000617001030 7	6
я	01525625022365014470044304000064 9	18
	78978758838689998987767851121826 0	138

ЛАБОРАТОРНАЯ РАБОТА № 11

СЕТЬ ФЕЙСТЕЛЯ

Цель работы: изучение принципа работы сети Фейстеля.

Описание лабораторной работы. В 1971 году Хорст Фейстель (*Horst Feistel*) запатентовал два устройства для реализации различных алгоритмов шифрования, получившие название «Люцифер» (*Lucifer*). Одно из устройств использовало конструкцию, впоследствии названную «сетью Фейстеля» (*Feistel cipher*, *Feistel network*). Работа над созданием новых криптосистем велась Фейстелем в стенах IBM вместе с Доном Копперсмитом (*Don Coppersmith*). Проект «Люцифер» был скорее экспериментальным, но стал базисом для алгоритма *DES* (*Data Encryption Standard*). В 1973 году Хорст Фейстель в журнале *Scientific American* опубликовал статью «Криптография и Компьютерная безопасность» (*Cryptography and Computer Privacy*), в которой раскрыл ряд важных аспектов шифрования и привел описание первой версии проекта «Люцифер».

Описание алгоритма. Сеть Фейстеля подразумевает разбиение обрабатываемого блока данных на несколько субблоков (чаще всего — на два), один из которых обрабатывается некоей функцией $f(\alpha)$ и накладывается на один или несколько остальных субблоков. На рисунке 3.4 приведена наиболее часто встречающаяся структура алгоритмов на основе сети Фейстеля.

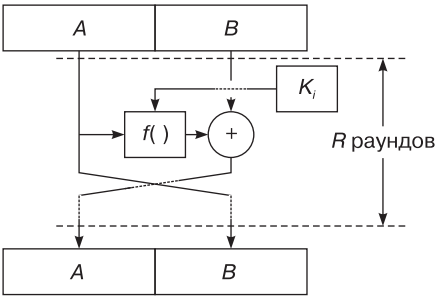


Рис. 3.4. Сеть Фейстеля

Блок открытого текста делится на две равные части A — левая (L) и B — правая (R), в каждом раунде i ($i = 1 \ldots n$ — номер раунда) вычисляется

$$L_i = R_{i-1} \oplus f(L_{i-1}, K_{i-1});$$
$$R_{i-1} = L_{i-1},$$

где $f()$ — некоторая функция, а K_{i-1} — ключ i-го раунда.

Результатом выполнения n раундов является (L_n, R_n), но обычно в n-ом раунде перестановка L_n и R_n не производится, что позволяет использовать ту же процедуру и для расшифрования, просто инвертировав порядок использования раундовой ключевой информации:

$$L_{i-1} = R_i \oplus f(L_i, K_{i-1});$$
$$R_{i-1} = L_i.$$

Одно из преимуществ такой модели — обратимость алгоритма независимо от используемой функции f, причем она может быть сколь угодно сложной.

Надежность алгоритма. Сети Фейстеля были широко изучены криптографами в силу их обширного распространения. В 1988 году Майкл Люби (*Michael Luby*) и Чарльз Ракофф (*Charles Rackoff*) провели исследования сети Фейстеля и доказали, что если раундовая функция является криптостойкой псевдослучайной и используемые ключи независимы в каждом раунде, то трех раундов будет достаточно для того, чтобы блочный шифр являлся псевдослучайной перестановкой.

Иногда сеть Фейстеля в западной литературе называют *Luby — Rackoff block cipher* в честь Люби и Ракоффа, которые проделали большой объем теоретических исследований в этой области.

В дальнейшем, в 1997 г., Мони Наор (*Moni Naor*) и Омер Рейнголд (*Omer Reingold*) предложили упрощенный вариант конструкции Люби — Ракоффа из четырех раундов, где в качестве первого и последнего раунда используются две попарно независимые перестановки. Два средних раунда конструкции Наора — Рейнголда идентичны раундам в конструкции Люби — Ракоффа.

Во многих блочных шифрах на основе сети Фейстеля были найдены те или иные уязвимости, однако в ряде случаев эти уязвимости являются чисто теоретическими и при нынешней производительности компьютеров использовать их на практике для взлома невозможно.

Симметричные криптоалгоритмы, использующие сеть Фейстеля. Так как большинство блочных шифров создано на основе сетей Фейстеля, коротко упомянем некоторые из них.

DES

DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняются 16 этапов одинаковых действий, определяемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной).

На каждом этапе биты ключа сдвигаются и затем из 56 битов ключа выбираются 48 битов для цикловых ключей. Правая половина данных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через восемь *S*-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией *f*. Затем результат функции *f* объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 циклов.

FEAL

В качестве входа процесса шифрования используется 64-битовый блок открытого текста. Сначала блок данных подвергается операции XOR с 64 битами ключа. Затем блок данных расщепляется на левую и правую половины. Объединение левой и правой половин с помощью XOR образует новую правую половину. Левая половина и новая правая половина проходят через *n* этапов (первоначально четыре). На каждом этапе правая половина объединяется с помощью функции *f* с шестнадцатью битами ключа, и с помощью XOR — с левой половиной, создавая новую правую половину. Исходная правая половина (на начало этапа) становится новой левой половиной. После *n* этапов (левая и правая половины не переставляются после *n*-го этапа) левая половина снова объединяется с помощью XOR с правой половиной, образуя новую правую половину, затем левая и правая соединяются вместе в 64-битовое целое. Блок данных объединяется с помощью XOR с другими 64 битами ключа, и алгоритм завершается.

Функция *f* использует 32 бита данных и 16 битов ключа и смешивает их вместе. Сначала блок данных разбивается на 8-битовые подблоки, которые затем объединяются с помощью XOR и меняются местами.

RC2

RC2 — это шифр с 64-битовым блоком и переменной длиной ключа, разработанный как альтернатива DES. В соответствии с утверждениями компании *RSA Data Security* программные реализации RC2 в три раза быстрее DES. Алгоритм может использовать ключ переменной длины, от 0 байтов до максимальной длины строки, поддерживаемой компьютерной системой, причем скорость шифрования не зависит от размера ключа. Этот ключ предварительно используется для заполнения 128-байтовой таблицы, зависящей от ключа. RC2 не использует *S*-блоков, здесь используются две операции — «смешивание» и «перемешивание» (*mix* и *mash*), для каждого этапа выбирается одна из них.

RC2 не является итеративным блочным шифром. Это предполагает, что RC2 более устойчив к дифференциальному и линейному криптоанализу, чем другие блочные шифры, безопасность которых опирается на копировании схемы DES.

ГОСТ-28147—89

ГОСТ является 64-битовым алгоритмом с 256-битовым ключом. ГОСТ также использует дополнительный ключ в виде восьми различных *S*-блоков, функционирование которых рассматривается ниже. В процессе шифрования на 32 этапах последовательно выполняется алгоритм шифрования Фейстеля.

Для шифрования текст сначала разбивается на левую половину L и правую половину R. На этапе i алгоритма ГОСТ используется подключ K_i и выполняются следующие действия:

$$L_i = R_{i-1};$$
$$R_i = L_{i-1} f(R_{i-1}, K_i).$$

Функция f проста. Сначала правая половина и i-й подключ складываются по модулю 2^{32}. Результат разбивается на восемь четырехбитовых подблоков, каждый из которых поступает на вход своего *S*-блока. ГОСТ использует восемь различных *S*-блоков, первые четыре бита попадают в первый *S*-блок, вторые четыре бита — во второй *S*-блок, и т.д. Каждый *S*-блок представляет собой перестановку чисел от 0 до 15.

В этом случае, если на входе *S*-блока 0, то на выходе 7. Если на входе 1, на выходе 10, и т.д. Все восемь *S*-блоков в ГОСТ-28147—89 различны, они фактически являются дополнительным ключевым материалом. *S*-блоки должны храниться в секрете.

Выходы всех восьми *S*-блоков объединяются в 32-битовое слово, затем все слово циклически сдвигается влево на 11 битов. Наконец результат объединяется с помощью XOR с левой половиной и получается новая правая половина, а правая половина становится новой левой половиной. Количество таких циклов — 32.

Генерация подключей проста. 256-битовый ключ разбивается на восемь 32-битовых блоков: $k1$, $k2$, ... $k8$. На каждом этапе используется свой подключ. Расшифование выполняется также как и шифрование, но инвертируется порядок подключей k_i.

ГОСТ-28147—89 не определяет способ генерации *S*-блоков, говорится только, что блоки должны быть предоставлены каким-то образом. Это породило домыслы о том, что советский производитель может поставлять хорошие *S*-блоки «хорошим» организациям и плохие *S*-блоки тем организациям, которых производитель собирается надуть,

вследствие чего неофициальные переговоры с российским производителем микросхем ГОСТ выявили другую альтернативу. Производитель создает перестановки *S*-блока самостоятельно с помощью генератора случайных чисел.

Программная реализация сети Фейстеля

```
/* функция преобразования подблока по ключу (зависит
от конкретного алгоритма)
subblock — преобразуемый подблок
key — ключ
возвращаемое значение — преобразованный блок*/
int f (int subblock, int key);
/* Шифрование открытого текста
left — левый входной подблок
right — правый входной подблок
* key — массив ключей (по ключу на раунд)
rounds — количество раундов*/
void crypt (int *left, int *right, int rounds, int *key)
{
     int i, temp;
     for (i = 0; i < rounds; i++)
     {
        temp = *right ^ f (*left, key[i]);
        *right = *left;
        *left = temp;
     }
}
/* Расшифрование текста
left — левый зашифрованный подблок
right — правый зашифрованный подблок*/
void decrypt (int *left, int *right, int rounds, int *key)
{
     int i, temp;
     for (i = rounds — 1; i > = 0; i--)
     {
        temp = *left ^ f (*right, key [i]);
        *left = *right;
        *right = temp;
     }
}
```

Описание программного интерфейса. Главное окно программы представлено на рис. 3.5.

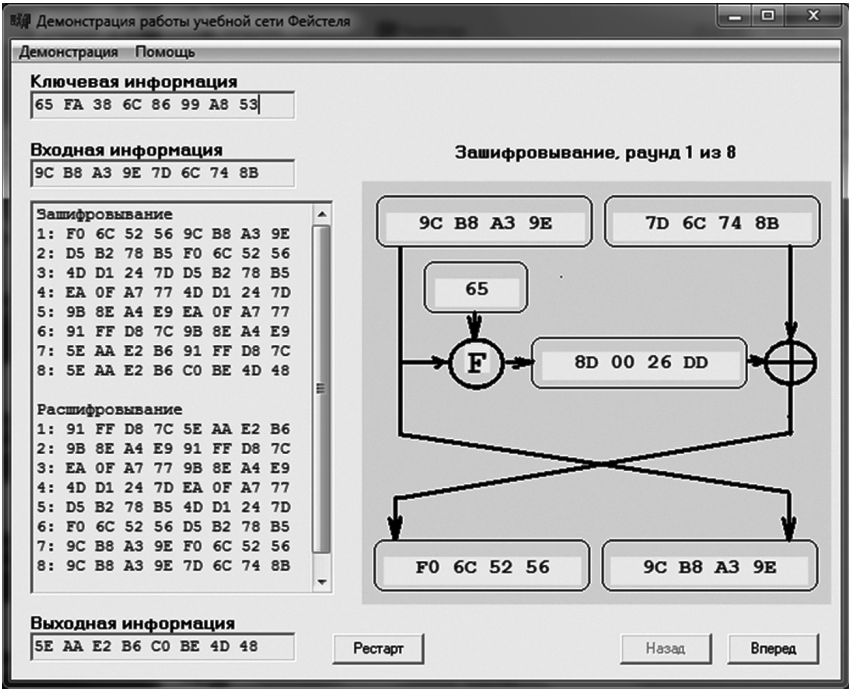


Рис. 3.5. Главное окно программы

Демонстрация

Рестарт — генерация случайного ключа (64 бит) и входной информации (64 бит).

Шаг вперед — переход к следующему раунду.

Шаг назад — переход к предыдущему раунду.

Выход — выход из программы.

Для удобства работы с программой основные функции продублированы кнопками в основном окне программы.

Помощь

Справка — содержит теоретическую информацию об алгоритме шифрования сеть Фейстеля и об использовании программы (рис. 3.6). Информация о разработчике представлена на рис. 3.7.

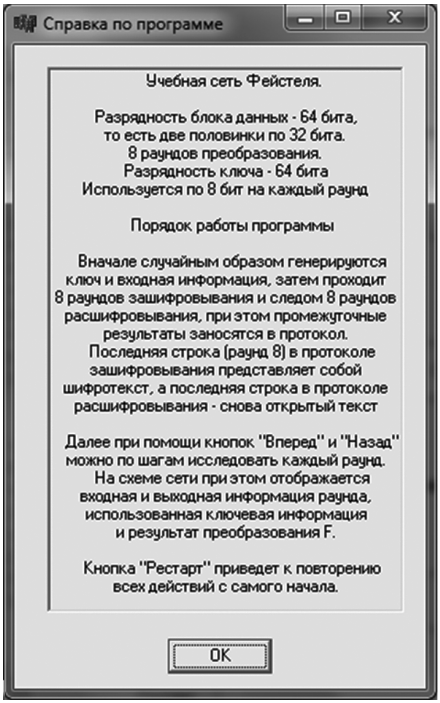


Рис. 3.6. Справочная информация

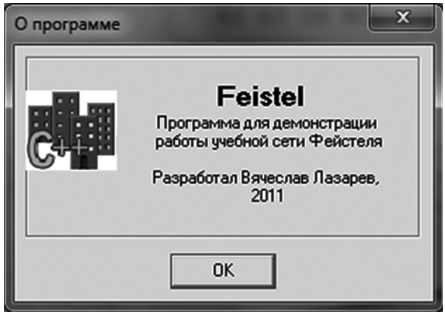


Рис. 3.7. Информация о разработчике

Поле «**Ключевая информация**» содержит случайно сгенерированный ключ.

Поле «**Входная информация**» содержит случайно сгенерированную информацию длиной 64 бита.

Поле **«Выходная информация»** содержит зашифрованную информацию.

Поле **«Зашифровывание»** отображает процесс шифрования, состоящий из восьми последовательных раундов.

Порядок выполнения программы

Нажать кнопку **«Рестарт»**, далее, используя клавиши вперед (назад), ознакомиться с процессом шифрования входной информации.

Задание

Для выполнения лабораторной работы на компьютере необходимо установить программу *Feistei. exe*. Запустить программу *Feistei. exe*, используемую для демонстрации работы сети Фейстеля.

1. Чтобы начать работу с программой, необходимо в Меню **«Демонстрация»** открыть вкладку **«Рестарт»** или нажать кнопку **«Рестарт»**.

2. С помощью переключателя **«Шаг вперед»** или кнопки **«Вперед»** можно последовательно наблюдать, каким образом шифруется на каждом цикле преобразования сети Фейстеля блок исходного текста.

3. Сохранить в отчете экранные формы, демонстрирующие процесс пошаговой работы сети Фейстеля при шифровании исходного текста.

4. Произвести вручную процесс шифрования в одном цикле и убедиться в том, что результаты, демонстрируемые программой, совпадают с произведенными расчетами. Сделать выводы по проделанной работе.

5. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 3.3.

Таблица 3.3

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	В каких современных симметричных системах шифрования используется сеть Фейстеля?
2, 4, 6, 8, 20, 22, 24, 26, 30	В чем отличие сбалансированной сети Фейстеля от несбалансированной сети Фейстеля? В каких блочных криптосистемах используется сбалансированная сеть?
11, 13, 15, 10, 17, 19, 27	В каких современных симметричных системах шифрования не используется сеть Фейстеля? Какие механизмы шифрования используются в этих криптографических системах?
12, 14, 16, 21, 23, 25, 29	Какой длины используются блоки для шифрования и цикловые ключи в блочных криптосистемах DES, FIAL, Blowfish, ГОСТ-28147—89?

ЛАБОРАТОРНАЯ РАБОТА № 12

РЕГИСТРЫ СДВИГА С ЛИНЕЙНОЙ ОБРАТНОЙ СВЯЗЬЮ КАК ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

Цель работы: изучение принципа работы генератора псевдослучайных последовательностей, основанного на регистре сдвига с линейной обратной связью.

Описание лабораторной работы

Общие сведения о регистрах сдвига с линейной обратной связью. Последовательности регистров сдвига используются как в криптографии, так и в теории кодирования. Их теория прекрасно проработана, потоковые шифры на базе регистров сдвига являлись рабочей лошадкой военной криптографии задолго до появления электроники.

Регистр сдвига с обратной связью (*далее РгССОС*) *состоит из двух частей*: *регистра сдвига и функции обратной связи* (рис. 3.8). Регистр сдвига представляет собой последовательность битов. Количество битов определяется ***длиной сдвигового регистра***, если длина равна ***n*** битам, то регистр называется ***n-битовым сдвиговым регистром***. Всякий раз, когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо на одну позицию. Новый крайний левый бит является функцией всех остальных битов регистра. На выходе сдвигового регистра оказывается один, обычно младший значащий, бит. ***Периодом сдвигового регистра*** называется длина получаемой последовательности до начала ее повторения.

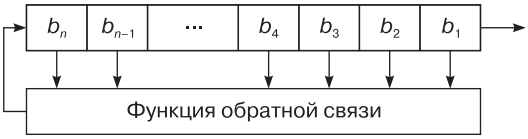


Рис. 3.8. Регистр сдвига с обратной связью

Регистры сдвига очень быстро нашли применение в потоковых шифрах, так как они легко реализовывались с помощью цифровой аппаратуры. В 1965 году Эрнст Селмер (*Ernst Selmer*), главный криптограф норвежского правительства, разработал теорию последовательности регистров сдвига. Соломон Голомб (*Solomon Golomb*), математик NSA, написал книгу, излагающие некоторые свои результаты и результаты Селмера. Простейшим видом регистра сдвига с обратной связью является *регистр сдвига с линейной обратной связью* (*linear feedback shift register*, *далее LFSR или РгССЛОС*). Обратная связь таких регистров

представляет собой просто XOR (сложение по модулю два) некоторых битов регистра, перечень этих битов называется отводной последовательностью (*tap sequence*). Иногда такой регистр называется конфигурацией Фиббоначи. Из-за простоты последовательности обратной связи для анализа РгССЛОС можно использовать довольно развитую математическую теорию. Проанализировав получаемые выходные последовательности, можно убедиться в том, что эти последовательности достаточно случайны, чтобы быть безопасными. РгССЛОС чаще других сдвиговых регистров используются в криптографии.

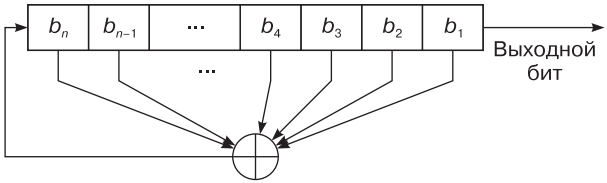


Рис. 3.9. РгССЛОС Фиббоначи

В общем случае n-битовый РгССЛОС может находиться в одном из $N = 2^n - 1$ внутренних состояний. Это означает, что теоретически такой регистр может генерировать псевдослучайную последовательность с периодом $T = 2^n - 1$ битов. (Число внутренних состояний и период равны $N = T_{\max} = 2^n - 1$, потому что заполнение РгССЛОС нулями приведет к тому, что сдвиговый регистр будет выдавать бесконечную последовательность нулей, что абсолютно бесполезно.) Только при определенных отводных последовательностях РгССЛОС циклически пройдет через все $2^n - 1$ внутренних состояний, такие РгССЛОС являются ***РгССЛОС с максимальным периодом***. Получившийся результат называется ***М-последовательностью***.

Пример. На рисунке 3.10 показан четырехбитовый РгССЛОС с отводом от первого и четвертого битов. Если его проинициализировать значением 1111, то до повторения регистр будет принимать следующие внутренние состояния (табл. 3.4).

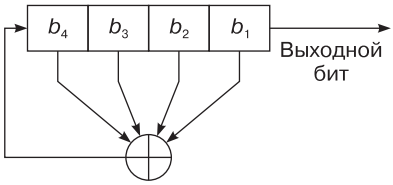


Рис. 3.10. Четырехбитовый РгССЛОС с отводом от первого и четвертого битов

Таблица 3.4

Номер такта сдвига (внутреннего состояния)	Состояние регистров				Выходной бит
	T_1	T_2	T_3	T_4	
Инициальное значение	**1**	**1**	**1**	**1**	**—**
1	0	1	1	1	1
2	1	0	1	1	1
3	0	1	0	1	1
4	1	0	1	0	0
5	1	1	0	1	1
6	0	1	1	0	0
7	0	0	1	1	1
8	1	0	0	1	1
9	0	1	0	0	0
10	0	0	1	0	0
11	0	0	0	1	1
12	1	0	0	0	0
13	1	1	0	0	0
14	1	1	1	0	0
15 (возврат в инициальное состояние)	**1**	**1**	**1**	**1**	**1**
16 (повтор состояний)	**0**	**1**	**1**	**1**	**1**

Выходной последовательностью будет строка младших значащих битов: 111101011001000 с периодом $T = 15$, общее число возможных внутренних состояний (кроме нулевого) $N = 2^4 - 1 = 16 - 1 = 15 = T_{\max}$, следовательно, выходная последовательность — M-последовательность.

Для того чтобы конкретный РгССЛОС имел максимальный период, многочлен, образованный из отводной последовательности и константы 1, должен быть примитивным по модулю 2. Многочлен представляется в виде суммы степеней, например многочлен степени n представляется так:

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0 = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0,$$

где $a_i = \{0,1\}$ для $i = 1 \ldots n$, а x^i — указывает разряд.

Степень многочлена является длиной сдвигового регистра. Примитивный многочлен степени n — это неприводимый многочлен, который является делителем $x^{2^n-1} + 1$, но не является делителем $x^d + 1$ для всех d, являющихся делителями $2^n - 1$.

В общем случае не существует простого способа генерировать примитивные многочлены данной степени по модулю 2. Проще всего вы-

бирать многочлен случайным образом и проверять, не является ли он примитивным. Это нелегко и чем-то похоже на проверку, не является ли простым случайно выбранное число — но многие математические пакеты программ умеют решать такую задачу.

Некоторые, но, конечно же не все, многочлены различных степеней, примитивные по модулю 2, приведены далее. Например, запись (32, 7, 5, 3, 2, 1, 0) означает, что следующий многочлен примитивен по модулю 2: $x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1$.

Это можно легко обобщить для РгССЛОС с максимальным периодом. Первым числом является длина РгССЛОС. Последнее число всегда равно 0, и его можно опустить. Все числа, за исключением 0, задают отводную последовательность, отсчитываемую от левого края сдвигового регистра. Другими словами, члены многочлена с меньшей степенью соответствуют позициям ближе к правому краю регистра.

Продолжая пример, запись (32, 7, 5, 3, 2, 1, 0) означает, что для взятого 32-битового сдвигового регистра новый бит генерируется с помощью XOR тридцать второго, седьмого, пятого, третьего, второго и первого битов, получающийся РгССЛОС будет иметь максимальную длину, циклически проходя до повторения через $2^{32} - 1$ значений.

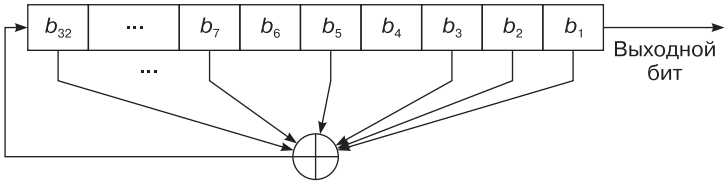


Рис. 3.11. 32-битовый РгССЛОС с максимальной длиной

Рассмотрим программный код РгССЛОС, у которого отводная последовательность характеризуется многочленом (32, 7, 5, 3, 2, 1, 0). На языке *C* это выглядит следующим образом:

```
int LFSR ()
{
     static unsigned long ShiftRegister = 1;
     /* Все, кроме 0. */
     ShiftRegister = ((((ShiftRegister >> 31)
     ^(ShiftRegister >> 6)
     ^(ShiftRegister >> 4)
     ^(ShiftRegister >> 2)
     ^(ShiftRegister >> 1)
```

```
      ^ShiftRegister))
      &0x00000001)
      <<31)
      |(ShiftRegister >> 1);
      return ShiftRegister & 0x00000001;
}
```

Если сдвиговый регистр длиннее компьютерного слова, код усложняется, но не намного. В приложении В приведена таблица некоторых примитивных многочленов по модулю 2, будем использовать ее в дальнейшем для выявления некоторых свойств этих многочленов, а также в программной реализации для задания отводной последовательности.

Следует обратить внимание, что у всех элементов таблицы нечетное число коэффициентов. Такая длинная таблица приведена для дальнейшей работы с РгССЛОС, так как РгССЛОС часто используются для работы с потоковыми шифрами и в генераторах псевдослучайных чисел. В нашем случае можно использовать многочлены со старшей степенью не более семи.

Если $p(x)$ примитивен, то примитивен и $x^n p(1/x)$, поэтому каждый элемент таблицы на самом деле определяет два примитивных многочлена. Например, если (a, b, 0) примитивен, то примитивен и (a, a-b, 0). Если примитивен (a, b, c, d, 0), то примитивен и (a, a-d, a-c, a-b, 0). Математически:

если примитивен $x^a + x^b + 1$, *то примитивен и* $x^a + x^{a-b} + 1$,
если примитивен $x^a + x^b + x^c + x^d + 1$,
то примитивен и $x^a + x^{a-d} + x^{a-c} + x^{a-b} + 1$.

Наиболее просто программно реализуются примитивные трехчлены, так как для генерации нового бита нужно выполнять XOR только двух битов сдвигового регистра (нулевой член не учитывается, т.е. $x^0 = 1$, см. пример выше). Действительно, все многочлены обратной связи, приведенные в таблице, являются разреженными, т.е., у них немного коэффициентов. Разреженность всегда представляет собой источник слабости, которой иногда достаточно для вскрытия алгоритма. Для криптографических алгоритмов гораздо лучше использовать плотные примитивные многочлены, у которых много коэффициентов. Применяя плотные многочлены, особенно в качестве части ключа, можно использовать значительно более короткие РгССЛОС.

Генерировать плотные примитивные многочлены по модулю 2 нелегко. В общем случае для генерации примитивных многочленов степени k нужно знать разложение на множители числа $2^k - 1$.

Сами по себе РгССЛОС являются хорошими генераторами псевдослучайных последовательностей, но они обладают некоторыми нежелательными неслучайными (детерминированными) свойствами. Последовательные биты линейны, что делает их бесполезными для шифрования. Для РгССЛОС длины *n* внутреннее состояние представляет собой предыдущие *n* выходных битов генератора. Даже если схема обратной связи хранится в секрете, она может быть определена по 2*n* выходным битам генератора с помощью высокоэффективного алгоритма *Berlekamp — Massey*.

Кроме того, большие случайные числа, генерируемые с использованием идущих подряд битов этой последовательности, сильно коррелированны и для некоторых типов приложений вовсе не являются случайными. Несмотря на это, РгССЛОС часто используются в качестве составных частей систем и алгоритмов шифрования.

О потоковых шифрах на базе РгССЛОС. Основной подход при проектировании генератора потока ключей на базе РгССЛОС прост. Сначала берется один или несколько РгССЛОС обычно с различными длинами и различными многочленами обратной связи. Если длины взаимно просты, а все многочлены обратной связи примитивны, то у образованного генератора будет максимальная длина. Ключ является начальным состоянием регистров РгССЛОС. Каждый раз для получения нового бита, достаточно сдвинуть на бит регистры РгССЛОС (это иногда называют тактированием (*clocking*)). Бит выхода представляет собой функцию, желательно нелинейную, некоторых битов регистров РгССЛОС. Эта функция называется комбинирующей функцией, а генератор в целом — комбинационным генератором. Если бит выхода является функцией единственного РгССЛОС, то генератор называется фильтрующим генератором. Большая часть теории подобного рода устройств разработана Селмером (*Selmer*) и Нилом Цирлером (*Neal Zierler*). Можно ввести ряд усложнений. В некоторых генераторах для различных РгССЛОС используется различная тактовая частота, иногда частота одного генератора зависит от выхода другого. Все это электронные версии идей шифровальных машин, появившихся до Второй мировой войны, которые называются генераторами с управлением тактовой частотой (*clock-controlled generators*). Управление тактовой частотой может быть с прямой связью, когда выход одного РгССЛОС управляет тактовой частотой другого РгССЛОС, или с обратной связью, когда выход одного РгССЛОС управляет его собственной тактовой частотой. Хотя все эти генераторы чувствительны, по крайней мере теоретически, к вскрытиям вложением и вероятной корреляцией, многие из них безопасны до сих пор.

Ян Касселлс (*Ian Cassells*), ранее возглавлявший кафедру чистой математики в Кембридже и работавший криптоаналитиком в Блетчли Парк (*Bletchly Park*), сказал, что «криптография — это смесь математики и путаницы, и без путаницы математика может быть использована против вас». Он имел в виду, что в потоковых шифрах для обеспечения максимальной длины и других свойств необходимы определенные математические структуры, такие как РгСсЛОС, но, чтобы помешать кому-либо получить содержание регистра и вскрыть алгоритм, необходимо внести некоторый сложный нелинейный беспорядок. Этот совет справедлив и для блочных алгоритмов.

Большинство реальных потоковых шифров основаны на РгСсЛОС. Даже в первые дни электроники построить их было несложно. Сдвиговый регистр не представляет собой ничего большего, чем массив битов, а последовательность обратной связи — набор вентилей XOR. Даже при использовании современных интегральных схем потоковый шифр на базе РгСсЛОС обеспечивает немалую безопасность с помощью нескольких логических вентилей. Проблема РгСсЛОС состоит в том, что их программная реализация очень неэффективна. Вам приходится избегать разреженных многочленов обратной связи — они облегчают корреляционные вскрытия — а плотные многочлены обратной связи неэффективны.

Эта отрасль криптографии быстро развивается и находится под зорким государственным контролем со стороны NSA. Большинство разработок засекречены — множество используемых сегодня военных систем шифрования основаны на РгСсЛОС. Действительно, у большинства компьютеров *Cray* (*Cray* 1, *Cray X-MP*, *Cray Y-MP*) есть весьма любопытная инструкция, обычно называемая как «счетчик совокупности» (*population count*). Она подсчитывает количество единиц в регистре и может быть использована как для эффективного вычисления расстояния Хэмминга между двумя двоичными словами так и для реализации векторизированной версии РгСсЛОС. Эта инструкция считается канонической инструкцией NSA, обязательно фигурирующей почти во всех контрактах, касающихся компьютеров.

С другой стороны, было взломано удивительно большое число казавшихся сложными генераторов на базе сдвиговых регистров.

О линейной сложности генерируемой последовательности псевдослучайных чисел РгСсЛОС. Анализировать потоковые шифры часто проще, чем блочные. Например, важным параметром, используемым для анализа генераторов на базе РгСсЛОС, является линейная сложность (*linear complexity*), или линейный интервал. Она определяется как длина n самого короткого РгСсЛОС, который может имитировать выход

генератора. Любая последовательность, сгенерированная конечным автоматом над конечным полем, имеет конечную линейную сложность. Линейная сложность важна, потому что с помощью простого алгоритма, называемого алгоритмом *Berlekamp-Massey*, можно определить этот РгССЛОС, проверив только 2*n* битов потока ключей. Воссоздавая нужный РгССЛОС, вы взламываете потоковый шифр.

Эта идею можно расширить с полей на кольца и на случаи, когда выходная последовательность рассматривается как числа в поле нечетной характеристики. Дальнейшее расширение приводит к вводу понятия профиля линейной сложности, который определяет линейную сложность последовательности по мере ее удлинения. Существуют также понятия сферической и квадратичной сложности. В любом случае следует помнить, что высокая линейная сложность не обязательно гарантирует безопасность генератора, но низкая линейная сложность указывает на недостаточную безопасность генератора.

О корреляционной независимости генерируемой последовательности псевдослучайных чисел РгССЛОС. Криптографы пытаются получить высокую линейную сложность, нелинейно объединяя результаты некоторых выходных последовательностей. При этом опасность состоит в том, что одна или несколько внутренних выходных последовательностей — часто просто выходы отдельных РгССЛОС — могут быть связаны общим ключевым потоком и вскрыты при помощи линейной алгебры. Часто такое вскрытие называют корреляционным вскрытием, или вскрытием разделяй-и-властвуй. Томас Сигенталер (*Thomas Siegenthaler*) показал, что можно точно определить корреляционную независимость и что существует компромисс между корреляционной независимостью и линейной сложностью.

Основной идеей корреляционного вскрытия является обнаружение некоторой корреляции между выходом генератора и выходом одной из его составных частей. Тогда, наблюдая выходную последовательность, можно получить информацию об этом промежуточном выходе. Используя эту информацию и другие корреляции, можно собирать данные о других промежуточных выходах до тех пор, пока генератор не будет взломан.

Против многих генераторов потоков ключей на базе РгССЛОС успешно использовались корреляционные вскрытия и их вариации, такие как быстрые корреляционные вскрытия, предлагающие компромисс между вычислительной сложностью и эффективностью.

О других способах вскрытия генерируемой последовательности псевдослучайных чисел РгССЛОС. Существуют и другие способы вскрытия генераторов потоков ключей. Тест на линейную корректность (*linear*

consistency) — это попытка найти некоторое подмножество ключа шифрования с помощью матричной техники. Существует и вскрытие корректности «встречей посередине» (*meet-in-the-middle consistency attack*). Алгоритм линейного синдрома (*linear syndrome algorithm*) основан на возможности записать фрагмент выходной последовательности в виде линейного уравнения. Существует вскрытие лучшим аффинным приближением (*best affine approximation attack*) и вскрытие выведенным предложением (*derived sequence attack*). К потоковым шифрам можно применить также методы дифференциального и линейного криптоанализа.

На рисунке 3.12 приведена обобщенная схема алгоритма РгССЛОС, рассматриваемого в лабораторной работе.

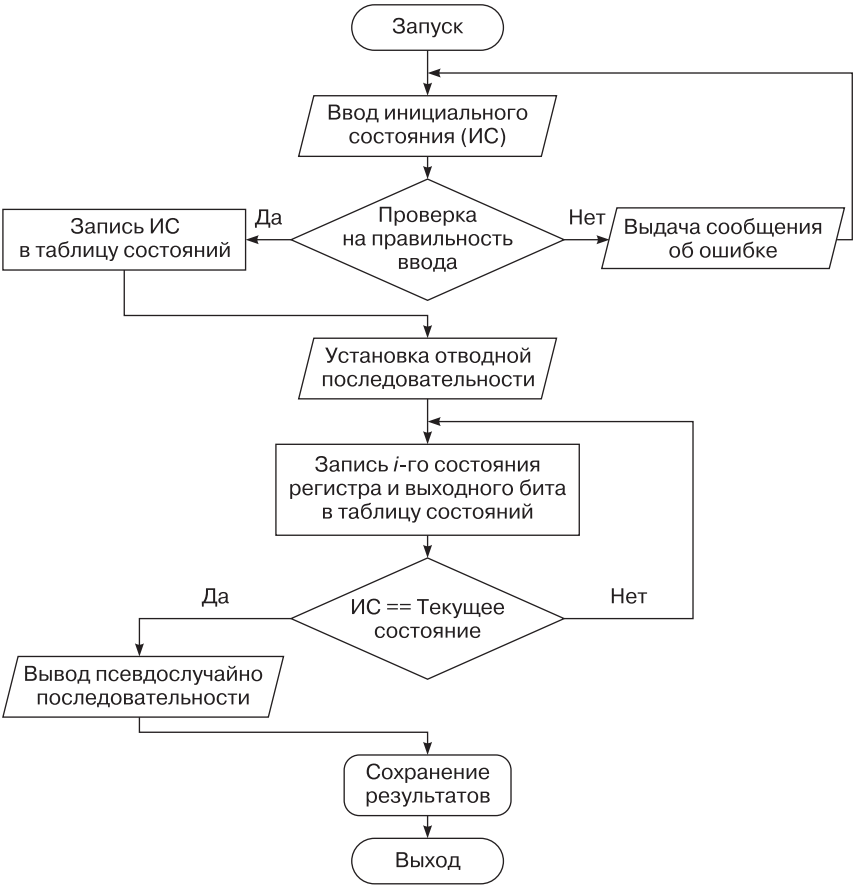


Рис. 3.12. Обобщенная схема алгоритма РгССЛОС

Описание программного интерфейса. На рис. 3.13 представлено главное окно программы.

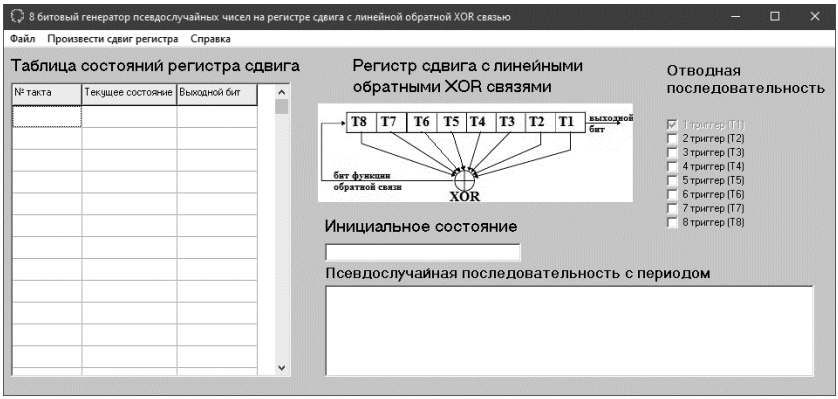


Рис. 3.13. Главное окно программы

В меню есть следующие функции:

- *Файл → Сохранить отчет*

Эта функция осуществляет создание файла отчета, куда записывается инициальное состояние РгССЛОС, отводная последовательность, период полученной псевдослучайной последовательности бит, сама последовательность и таблица состояний. Если файл успешно был сохранен, то выдается сообщение об успешном сохранении (рис. 3.14). Рекомендуемое расширение файла отчета *.txt. Если сохранение отчета выполняется до внесения значений в поля формы, то выдается сообщение об ошибке (рис. 3.15).

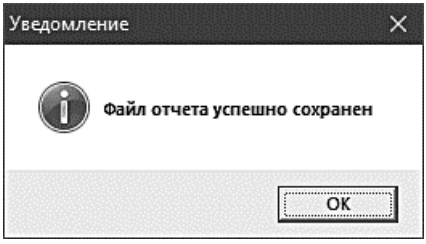


Рис. 3.14. Уведомление об успешном сохранении файла отчета

- *Файл → Выход*

Эта функция обеспечивает закрытие приложения.

- *Произвести сдвиг регистра*

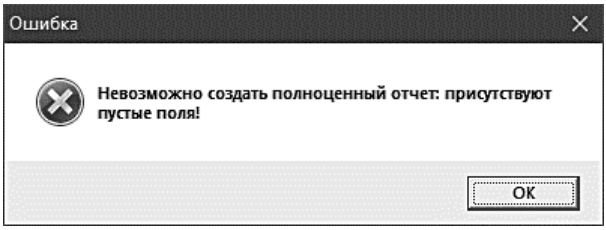


Рис. 3.15. Уведомление об ошибке создания отчета

Эта функция эмулирует работу регистра сдвига. Последовательно производя 256 сдвигов, каждый сдвиг формирует выходной бит псевдослучайной последовательности и новое состояние регистра. Как только находится состояние регистра идентичное инициальному, вычисляется период и выводится в поле «Псевдослучайная последовательность с периодом» полученная псевдослучайная последовательность. В программном инструментарии предусмотрена обработка ошибок ввода значения в поле «Инициальное состояние»:

1. Значение не задано (пустая строка) или задано, но его длина составляет менее 8 бит / 1 байта (рис. 3.16);

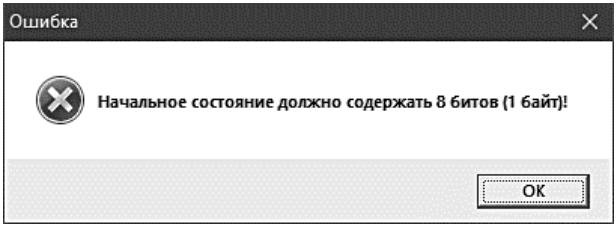


Рис. 3.16. Сообщение о том, что используется недопустимая длина начального состояния

2. Введенное значение состоит не только из нулей и единиц – не является двоичным (рис. 3.17);

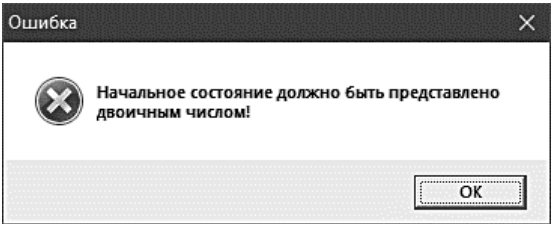


Рис. 3.17. Сообщение о том, что система счисления для указания начального состояния выбрана неверно

3. Введенное значение состоит только из нулей (рис. 3.18).

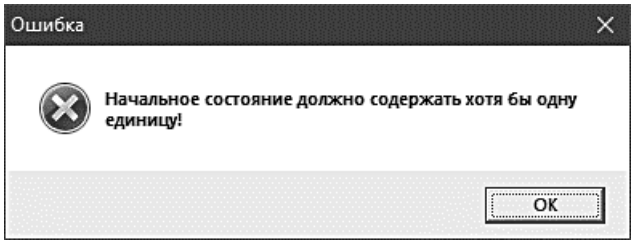


Рис. 3.18. Сообщение о том, что строка начального состояния не содержит единиц

■ *Справка → О программе*

Эта функция выводит на экран краткое описание программы и инструкцию (рис. 3.19).

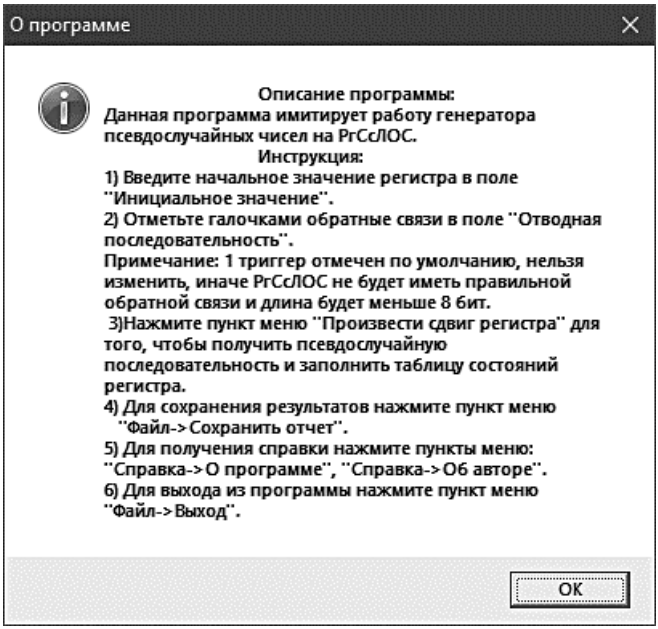


Рис. 3.19. О программе

■ *Справка → Об авторе*

Эта функция выводит на экран информацию об авторе программы (рис. 3.20).

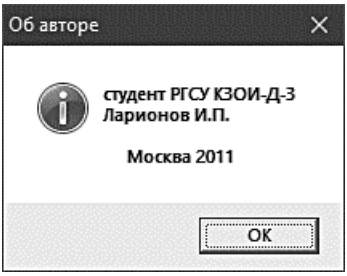


Рис. 3.20. Об авторе

Задание

1. Запустите программу 8*bit-LFSR.exe*. Ознакомьтесь со справкой по программе для выполнения работы (см. рис. 3.19).

2. Введите в поле «**Инициальное значение**» любое восьмиразрядное двоичное число.

3. Установите отводную последовательность (по умолчанию обязательно один триггер должен входить в отводную последовательность и должна присутствовать хотя бы одна обратная связь, иначе регистр будет неисправен).

4. Нажмите пункт меню «**Произвести сдвиг регистра**».

5. В отчете по лабораторной работе отразите результаты, полученные в п. 4 (**Файл — Сохранить отчет**):

- введенное в п. 2 инициальное значение;
- многочлен, характеризующий отводную последовательность;
- период полученной псевдослучайной последовательности;
- полученную псевдослучайную последовательность.

Ответьте на вопрос: является ли полученная в п. 5 последовательность максимальной? Почему?

6. Выполните п. 3—5 с теми же параметрами, но в отводной последовательности укажите только один триггер.

Ответьте на вопрос: какое значение поступает на вход восьмого триггера при каждом его сдвиге и как оно вычисляется в случае если присутствует одна обратная связь? Если присутствует две или более (до восьми) обратных связей?

7. Выполните п. 3—5 с теми же параметрами, но в качестве инициальной последовательности укажите «11111111».

Ответьте на вопрос: влияет ли инициальное значение на период псевдослучайной последовательности? Влияет ли инициальное значение на псевдослучайную последовательность?

- Измените отводную последовательность (должно быть минимум две связи), а инициальное значение оставьте тем же. Произведите сдвиг регистра. Что изменилось по сравнению с предыдущим пунктом?
- Измените отводную последовательность, установив только одну обратную связь T1. Произведите сдвиг регистра. Что изменилось по сравнению с предыдущим пунктом? Почему период последовательности отличается от аналогичных в предыдущих пунктах (п. 6—7)?

8. Выполните п. 3—5 с теми же параметрами, но в качестве инициальной последовательности укажите «00000000».

Ответьте на вопрос: почему это значение недопустимо при работе РгССЛОС?

9. Выполните п. 3—5, но в отводной последовательности отметьте галочками 1, 5, 6 и 7 триггеры или 1, 4, 6, 8 триггеры.

- Измените несколько раз инициальное значение и произведите сдвиг регистра.

Ответьте на вопросы: Каков период псевдослучайной последовательности? Почему он принимает это значение? Какой многочлен по модулю два задает такую последовательность? Является ли он приводимым? Что меняется при изменении инициального значения?

10. Предоставьте электронный отчет преподавателю. Отчет должен содержать ответы на вопросы п. 5—10, сведения для п. 5—10, описанные в п. 5, 1, и скриншот главной формы и формы «**Справка — О программе**» (п. 1).

11. Включите в отчет о лабораторной работе ответы на вопросы, выбранные в соответствии с номером варианта из табл. 3.5.

Таблица 3.5

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Что такое М-последовательность? Каковы ее свойства? Какая отводная последовательность позволяет ее получить? Опишите процесс работы четырехбитового РгССЛОС: откуда берется выходной бит и как формируется псевдослучайная последовательность, как происходит сдвиг регистра, как меняется его состояние, как образуется бит функции обратной связи (приведите таблицу истинности для операции XOR от 1, 2, 3, 4 переменных)
2, 4, 6, 8, 20, 22, 24, 26, 30	Что определяет свойство периодичности РгССЛОС? Отчего зависит период РгССЛОС? Какие многочлены являются неприводимыми по модулю 2? Где и для чего их можно применять на практике?
11, 13, 15, 10, 17, 19, 27	Что входит в понятие «Линейная сложность бинарной последовательности»? Как ее можно использовать для оценки псевдослучайно бинарной последовательности? Что определяет свойство периодичности РгССЛОС? Отчего зависит период РгССЛОС?
12, 14, 16, 21, 23, 25, 29	Что такое отводная последовательность? Для чего она нужна в РгССЛОС и на какие его параметры влияет? Что такое инициальное состояние РгССЛОС? Какие есть ограничения на значение инициального состояния? Почему? Что такое ГСПЧ? На чем они могут быть основаны? Какие у них преимущества и недостатки?

ЧАСТЬ 4

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Задача защиты информации от несанкционированного доступа решалась самыми разнообразными способами на различных этапах истории. Уже в древнем мире выделилось два основных направления решения этой задачи, существующие и по сегодняшний день: *криптография* и *стеганография*. Целью криптографии, как уже указывалось, является скрытие содержимого сообщений за счет их шифрования. В отличие от этого стеганография скрывает сам факт существования тайного сообщения.

Слово «*стеганография*» в переводе с греческого буквально означает «тайнопись» (*steganos* — секрет, тайна; *graphy* — запись). Тайнопись осуществляется самыми различными способами. Общей чертой этих способов является то, что скрываемое сообщение встраивается в некоторый безобидный, не привлекающий внимание объект. Затем этот объект открыто передается адресату, не вызывая какого-либо подозрения со стороны.

В различные времена и эпохи люди применяли самые разнообразные стеганографические методы для защиты своих секретов. Местом зарождения стеганографии многие называют Египет, хотя первыми «стеганографическими сообщениями» можно назвать и наскальные рисунки древних людей.

Первое упоминание о стеганографических методах в литературе приписывается Геродоту, описавшему случай передачи сообщения Демартом, который хотел послать в Спарту сообщение об угрозе нападения Ксерксов. Тогда он соскоблил воск с дощечки, написал послание непосредственно на дереве, затем вновь покрыл ее воском. В результате доска выглядела неиспользованной и без проблем прошла досмотр центурионов.

Еще один, весьма неожиданный способ сокрытия информации или условных знаков — татуировка на голове бритого посланца. Для передачи тайного сообщения голову раба обривали, наносили на кожу татуировку, и когда волосы отрастали и сообщение становилось неви-

димым, его отправляли с посланием. Для прочтения требовалось побрить голову гонца.

В Китае письма писали на полосках шелка. Поэтому для сокрытия сообщений полоски с текстом письма сворачивались в шарики, покрывались воском и затем глотались посыльными.

Темное средневековье породило не только инквизицию: усиление слежки привело к развитию как криптографии, так и стеганографии. В XV веке монах Тритемиус (1462—1516), занимавшийся криптографией и стеганографией, описал множество различных методов скрытой передачи сообщений. Позднее, в 1499 г., эти записи были объединены в книгу *Steganographia*.

К стеганографии также относится написание текстов невидимыми (симпатическими) чернилами, проявляющимися при нагревании. Яркий пример применения таких чернил — история с В.И. Лениным, который в тюрьме писал статьи в «Искру» молоком между строк книги.

Особое место в истории стеганографии занимают фотографические микроточки, которые сводили с ума спецслужбы США во время Второй мировой войны. Однако микроточки появились намного раньше, сразу же после изобретения Луи Жак Манде Дагером фотографического процесса, и впервые в военном деле были использованы во времена франко-прусской войны в 1870 г.

Распространение стеганографии во время Второй мировой войны и тотальная шпиономания вызвали появление многих цензурных ограничений. В США были запрещены к международной почтовой пересылке описания шахматных партий, инструкции по вязанию и шитью, вырезки из газет, детские рисунки. Запрещалось посылать телеграммы с указанием доставить определенный сорт цветов к определенной дате, а впоследствии американским и английским правительствами были запрещены вообще все международные телеграммы, касающиеся доставки и заказа цветов.

В настоящее время в связи с бурным развитием вычислительной техники и новых каналов передачи информации появились и новые стеганографические методы, в основе которых лежат особенности представления информации в компьютерных файлах и вычислительных сетях. Это дает возможность говорить о становлении нового направления — цифровой или компьютерной стеганографии.

Как и любой новый раздел науки, компьютерная стеганография, несмотря на большое количество открытых публикаций и ежегодные конференции, долгое время не имела единой терминологии. Только на конференции *Information Hiding*: *First Information Workshop*

в 1996 г. было предложено использовать единую стеганографическую терминологию.

Стеганографическая система или *стегосистема* — совокупность средств и методов, которые используются для формирования скрытого канала передачи информации. При построении стегосистемы должны учитываться следующие положения:

- противник имеет полное представление о стеганографической системе и деталях ее реализации, единственной информацией, которая остается неизвестной потенциальному противнику, является ключ, с помощью которого только его держатель может установить факт присутствия и содержание скрытого сообщения;
- если противник каким-то образом узнает о факте существования скрытого сообщения, это не должно позволить ему извлекать подобные сообщения из других объектов до тех пор, пока ключ хранится в тайне;
- потенциальный противник должен быть лишен каких-либо технических и иных преимуществ в распознавании или раскрытии содержания тайных сообщений.

В качестве встраиваемых данных может использоваться любая информация: текст, звук, графика и пр.

Контейнер — любая информация, предназначенная для сокрытия тайных сообщений. Под *пустым контейнером* понимается контейнер без встроенного сообщения; *заполненный контейнер* или *стегоконтейнер* содержит встроенную информацию.

Встроенное (*скрытое*) *сообщение* — это сообщение, встраиваемое в контейнер.

Стеганографический канал, или *стегоканал* — канал передачи стего.

Стегоключ, или *ключ* — секретный ключ, необходимый для сокрытия информации. В зависимости от количества уровней защиты (например, встраивание предварительно зашифрованного сообщения) в стегосистеме может быть один или несколько стегоключей.

По типу ключа стегосистемы аналогично криптосистемам можно подразделить на два типа: *системы с секретным* и *с открытым ключом*.

В стегосистеме с секретным ключом используется один ключ, который должен быть определен либо до начала обмена секретными сообщениями, либо передан по защищенному каналу.

В стегосистеме с открытым ключом для встраивания и извлечения сообщения используются разные ключи, которые различаются таким образом, что с помощью вычислений невозможно вывести секретный

ключ из открытого. Поэтому открытый ключ может передаваться свободно по незащищенному каналу связи. Такая схема хорошо работает и при взаимном недоверии отправителя и получателя.

Любая стегосистема должна отвечать следующим требованиям:

- свойства контейнера должны быть таковы, чтобы изменение невозможно было выявить при визуальном контроле; это требование определяет качество сокрытия внедряемого сообщения: для обеспечения беспрепятственного прохождения стегосообщения по каналу связи оно никоим образом не должно привлечь внимание атакующего;
- стегосообщение должно быть устойчиво к искажениям, в том числе и злонамеренным; в процессе передачи изображение (звук или другой контейнер) может претерпевать различные трансформации: масштабироваться, преобразовываться в другой формат, кроме того, оно может быть сжато, в том числе и с использованием алгоритмов сжатия с потерей данных;
- для сохранения целостности встраиваемого сообщения необходимо использование, например, кодов с обнаружением и исправлением ошибок;
- для повышения надежности встраиваемое сообщение должно быть продублировано.

В настоящее время можно выделить три тесно связанных между собой и имеющих одни корни направления стеганографии: *сокрытие данных*, *цифровые водяные знаки* и *заголовки*.

Сокрытие внедряемых данных — это основное направление компьютерной стеганографии. На его основе базируются все остальные разделы этой развивающейся науки. Если говорить о скрытии информации для ее секретной передачи, то в данном случае встраиваемое сообщение обычно имеет большой размер и размер контейнера в несколько раз должен превышать объем встраиваемых данных.

Цифровые водяные знаки (*ЦВЗ*) используются для защиты авторских или имущественных прав на цифровые изображения, фотографии или другие оцифрованные произведения искусства. Основными требованиями, которые предъявляются к таким встроенным данным, являются надежность и устойчивость к искажениям. ЦВЗ имеют небольшой объем, однако с учетом указанных выше требований для их встраивания используются более сложные методы, чем для встраивания просто сообщений или заголовков.

Заголовки используется в основном для маркирования изображений в больших электронных хранилищах (библиотеках) цифровых

изображений, аудио и видео файлов. Внедряемые заголовки имеют небольшой объем, а предъявляемые к ним требования минимальны: заголовки должны вносить незначительные искажения и быть устойчивы к основным геометрическим преобразованиям.

Среди основных причин наблюдающегося всплеска интереса к стеганографии можно выделить принятые в ряде стран ограничения на использование сильной криптографии, а также проблему защиты авторских прав на художественные произведения в цифровых глобальных сетях. Поэтому в ближайшее время можно ожидать совершенствования существующих и разработки новых методов и алгоритмов сокрытия передаваемой информации.

ЛАБОРАТОРНАЯ РАБОТА № 13

ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МЕТОДАМИ СТЕГАНОГРАФИИ

Цель работы: ознакомление с методами защиты исполняемых файлов. Изучение возможностей стеганографической защиты.*exe* файлов путем встраивания цифровых водяных знаков в пустое место, в конце секции файла.

Примечание. Для выполнения лабораторной работы на компьютере необходимо установить программу ***Filigrana.exe***

Описание лабораторной работы. *Защита программного обеспечения* (*ПО*) — комплекс мер, направленных на защиту ПО от несанкционированного приобретения, использования, распространения, модифицирования, изучения и воссоздания аналогов.

Разработка наиболее эффективного метода защиты для того или иного программного продукта в настоящее время становится одной из важных задач большинства программистов, которые занимаются разработкой специализированного, платного программного обеспечения, так как это позволяет им продавать свой интеллектуальный труд и исключить возможности его нелегального использования среди потребителей, говоря иными словами, никто не сможет использовать оригинальную, лицензионную копию определенной программы, предварительно не купив, не заплатив денег ее разработчику.

Затраты производителей на создание эффективного метода защиты их программных продуктов окупаются и компенсируют потенциальный ущерб, наносимый нелегальным копированием и использованием программ.

Существуют два основных способа защиты интеллектуальной собственности и, следовательно, самих программных продуктов:

1) *юридический* — данный способ защиты заключается в создании определенных законодательных актов, которые будут охранять интеллектуальную собственность (в нашем случае программные продукты) от нелегального использования;

2) *технический* — реализуется путем включения в программный продукт какого-либо из существующих методов защиты, который будет запрещать его нелегальное использование. По сравнению с юридическим способом защиты он является наиболее распространенным, так как он практичен и сравнительно недорогой в реализации.

Юридическая защита. Юридическая защита включает в себя такие методы как патентование, оформление авторских прав на интеллектуальную собственность и т.д. Также предусматривается возможность лицензирования программного продукта, так, например большинство программных продуктов поставляются вместе с лицензией, которая подтверждает право пользователя использовать этот программный продукт, т.е., покупая лицензионную копию программы, пользователь в некой мере производит покупку лицензии на право работы с ее копией.

Техническая защита. Существует множество технических методов защиты программных продуктов. Ниже будут рассмотрены наиболее интересные и актуальные из них.

Использование парольной защиты (*кода активации*). Этот вариант на сегодня является самым распространенным. Принцип действия защиты основан на идентификации и аутентификации пользователя ПО путем запроса у него дополнительных данных, это могут быть название фирмы и (или) имя и фамилия пользователя и его пароля либо только пароля. Эта информация может запрашиваться в различных ситуациях, например при старте программы, по истечении срока бесплатного использования ПО, при вызове процедуры регистрации или в процессе установки на компьютер пользователя. Процедура защиты проста в реализации и поэтому очень часто применяется производителями ПО. Она сводится к проверке правильности вводимого кода и запуске или незапуске ПО в зависимости от результатов проверки.

Защита через Интернет. Данный метод защиты схож с предыдущим, его отличие заключается в том, что проверка кода активации происходит на сервере производителя. Если ваша программа работает с сетью (это может быть менеджер закачки файлов, браузер, FTP-клиент и пр.), можно применить проверку введенного в программу кода через Интернет, используя базу данных пользователей на сайте программы. Однако проверять нужно какие-либо косвенные данные, чтобы исследователь не мог получить с сайта программы, например, базу данных с серийными номерами.

Сложность взлома этой защиты в том, что весьма непросто найти в коде программы место, где идет обращение к Интернет-серверу именно с целью проверки регистрационного кода.

Выполнение программ на стороне сервера. Данный метод защиты основан на технологии клиент-сервер, он позволяет предотвратить отсылку кода программы пользователям, которые будут с ней работать, так как сама программа хранится и выполняется на сервере, а пользователи, используя клиентскую часть этой программы, получают результаты ее выполнения (рис. 4.1).

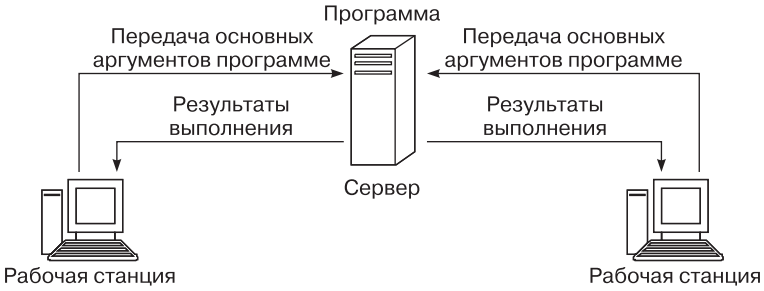


Рис. 4.1. Выполнение программы на стороне сервера

Клиентскую часть можно распространять бесплатно, это позволит пользователям платить только за использование серверной части.

Недостатком данного метода является то, что он устанавливает зависимость пропускной способности сети и тех данных, с которыми будет работать программа (соответственно работа с мультимедиа данными требует максимальной пропускной способности сети). Поэтому данный метод наиболее эффективен для простых программ (сценариев), потребность в которых очень велика среди пользователей.

Преимущество такого метода заключается в том, что злоумышленнику в данном случае нужно будет для начала скомпрометировать сам сервер, только после чего он сможет получить копию требуемой программы.

Данный метод не позволяет защитить программу от нелегального копирования, поэтому после того как ее копия попадет к злоумышленнику, он сможет делать с ней что захочет.

Установка подлинности кода (*tamper-proofing*). В данном случае в программу помещается процедура проверки целостности самой программы, что позволяет определить, была ли программа изменена (были ли внесены какие-либо изменения в ее код). Если эта процедура обнаруживает, что в программу внесены изменения, она делает программу нефункциональной (рис. 4.2).

Это позволяет защитить программный продукт от модификации со стороны злоумышленника.

Существуют такие пути проверки целостности программы, как:

- проверка идентичности оригинальной и запускаемой программы. Обычно для этого определяется контрольная сумма запущенной программы, которая потом сверяется с записанной в процедуру проверки контрольной суммой оригинальной программы; для осуществления быстрой проверки используют такие алгоритмы как CRC или MD4/5;

- проверка результатов работы программы, т.е. осуществляется проверка выходных значений функций, которые очень чувствительны, к каким либо возможным изменениям в программном коде;
- создание запускаемой программы на лету в соответствии с ее оригинальным образом — это позволяет избежать возможности выполнения изменений внесенных в программу, так как они не будут учитываться при ее создании.

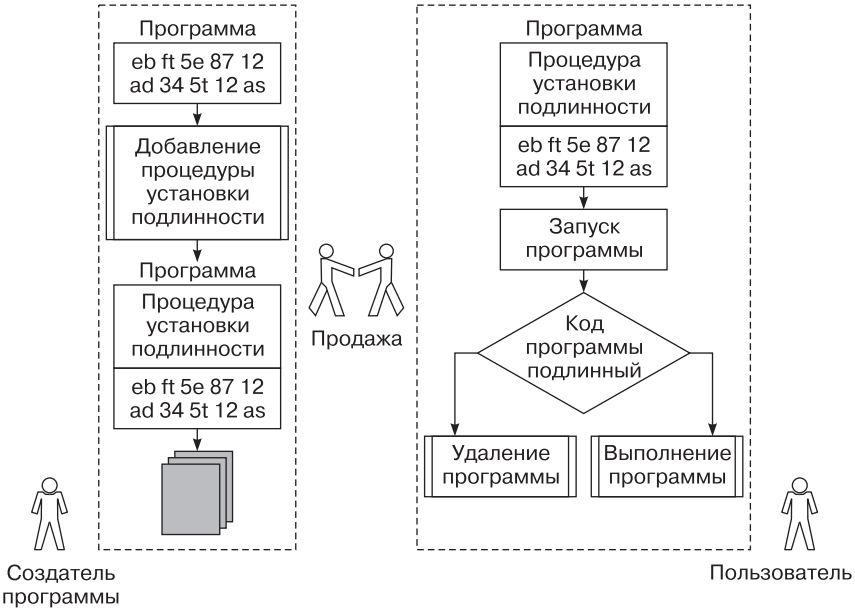


Рис. 4.2. Установка подлинности кода

Шифрование программного кода. Используется для того, чтобы предотвратить вмешательство в программу, а также усложнить процесс понимания взломщиком того, как устроена программа, как она работает, как в ней реализован метод защиты и т.д.

Данный метод защиты предусматривает шифрование кода программы, после чего она в зашифрованном виде поставляется конечным пользователям (иногда эффективно зашифровывать только наиболее важные, критические, участки кода, а не весь код программы). Когда пользователь запускает такую программу, вначале будет запущена процедура расшифровки программы, которой потребуется ключ, с помощью которого и будет расшифрована запускаемая программа (рис. 4.3).

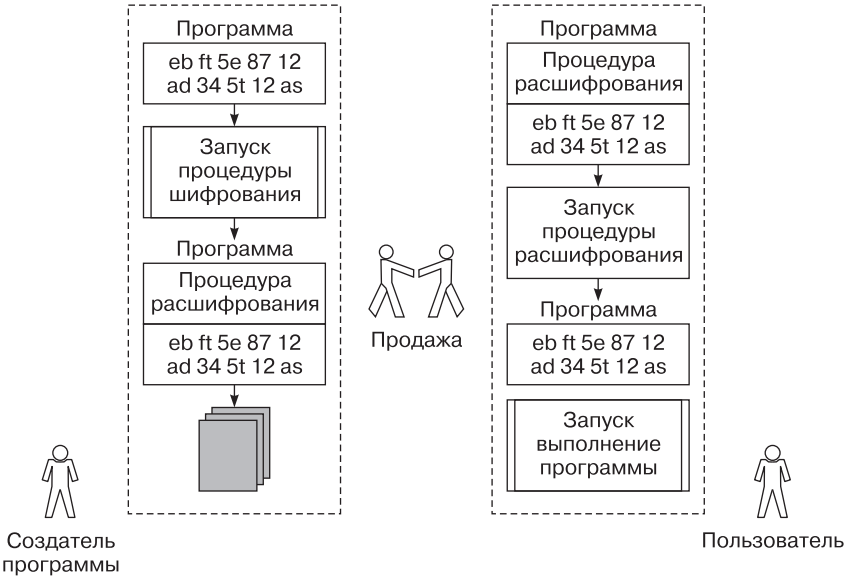


Рис. 4.3. Шифрование программного кода

Способ шифрования программ имеет недостатки, одним из которых является то, что у взломщика есть возможность после приобретения лицензионной копии программы произвести извлечение расшифрованных частей программы в процессе ее работы из памяти. Поэтому сразу после исполнения расшифрованного кода его необходимо выгружать из памяти.

Обфускация программного кода. Обфускация (*obfuscation* — запутывание), это один из методов защиты программного кода, который позволяет усложнить процесс реверсивной инженерии кода защищаемого программного продукта.

Суть процесса обфускации заключается в том, чтобы запутать программный код и устранить большинство логических связей в нем, т.е. трансформировать его так, чтобы он был очень труден для изучения и модификации посторонними лицами (будь то взломщики или программисты которые собираются узнать уникальный алгоритм работы защищаемой программы).

Использование только обфускации для обеспечения полной и эффективной защиты программных продуктов недостаточно, так как она не предоставляет возможности предотвращения нелегального использования программного продукта. Поэтому обфускацию обычно

используют вместе с одним из существующих методов защиты (например, шифрование программного кода), это позволяет значительно повысить уровень защиты программного обеспечения в целом (рис. 4.4).

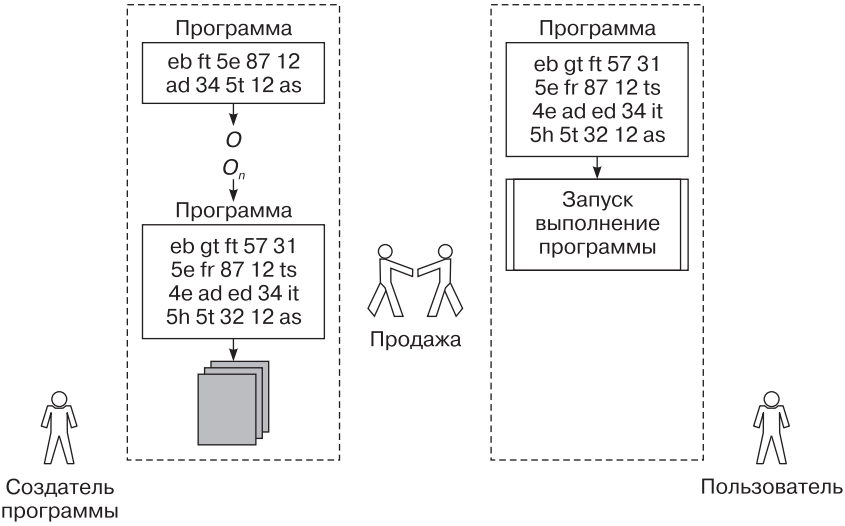


Рис. 4.4. Обфускация программного кода

Так как код, получаемый после осуществления обфускации над одной и той же программой, разный, то процесс обфускации можно использовать для быстрой локализации нарушителей авторских прав (т.е. тех покупателей, которые будут заниматься нелегальным распространением купленных копий программ). Для этого определяют контрольную сумму каждой копии программы, прошедшей обфускацию, и записывают ее вместе с информацией о покупателе в соответствующую базу данных. После этого для определения нарушителя достаточно будет, определив контрольную сумму нелегальной копии программы, сопоставить ее с информацией, хранящейся в базе данных.

Программы-протекторы. Протекторами называются программы, предназначенные для защиты программ от взлома. Данный способ защиты программы стал весьма популярным в последнее время. После того как получен работающий исполняемый файл, этот файл обрабатывается с помощью программы-протектора и создается новый исполняемый файл, в котором реализованы различные средства защиты

(защита программ представляет собой обычную упаковку всего файла, при этом «защищается» сам распаковщик, который в итоге распаковывает файл). Протекторы пишутся профессионалами в области защиты и обеспечивают неплохой уровень противодействия взлому, с которым большинству исследователей программ не справиться.

Недостатком данного метода является то, что протекторы становятся очень популярными и, соответственно, активно изучаются взломщиками, которые пишут программы-антипротекторы (распаковщики защиты). Подобные средства быстро и автоматически снимают защиту.

Использование электронных ключей. На сегодня это наиболее надежный и удобный метод защиты тиражируемого ПО средней и высшей ценовой категории. Он обладает высокой стойкостью к взлому и не ограничивает использование легальной копии программы.

Электронный ключ представляет собой небольшое устройство, которое подсоединяется к одному из портов компьютера (COM, LPT, USB). Принцип его действия состоит в следующем: ключ присоединяется к определенному интерфейсу компьютера, далее защищенная программа через специальный драйвер отправляет ему запрос, который обрабатывается в соответствии с заданным алгоритмом и возвращается обратно. Если ответ ключа правильный, то программа продолжает свою работу. В противном случае она может выполнять любые действия, заданные разработчиками (например, переключаться в демонстрационный режим, блокируя доступ к определенным функциям). Вот некоторые характерные запросы:

- проверка наличия подключения ключа;
- считывание с ключа необходимых программе данных в качестве параметра запуска;
- запрос на расшифрование данных или исполняемого кода, необходимых для работы программы (предварительно разработчик защиты шифрует часть кода программы, и при этом непосредственное выполнение такого зашифрованного кода приводит к ошибке);
- проверка целостности исполняемого кода путем сравнения его текущей контрольной суммы с оригинальной контрольной суммой, считываемой с ключа;
- запрос к встроенным в ключ часам реального времени (при их наличии) и т.д.

Специфика PE-файлов. Типичный исполняемый PE-файл имеет следующую структуру (рис. 4.5).

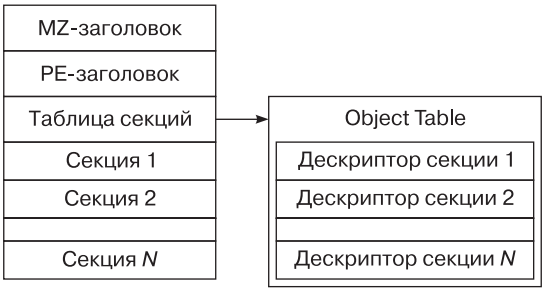


Рис. 4.5. Обобщенная структура PE-файла

MZ-заголовок содержит программу для ОС DOS — эта программа называется stub и нужна для совместимости со старыми ОС. Если мы запускаем PE-файл под ОС DOS или OS/2, она выводит на экран консоли текстовую строку, которая информирует пользователя, что данная программа не совместима с данной версией ОС: *This program cannot be run in DOS mode*. После этой DOS-программы идет структура, которая называется IMAGE_NT_HEADERS (PE-заголовок). Она описывается следующим образом:

```
typedef struct _IMAGE_NT_HEADERS
{
     DWORD Signature;
     IMAGE_FILE_HEADER FileHeader;
     IMAGE_OPTIONAL_HEADER32 OptionalHeader;
}
```

Первый элемент IMAGE_NT_HEADERS — сигнатура PE-файла, которая равна 0x00004550 или в ASCII-символах, «PE00». Кстати, именно из-за этой сигнатуры исполняемый файл Windows и называют PE-файлом.

Далее идет структура, которая называется файловым заголовком и определена как IMAGE_FILE_HEADER. Файловый заголовок содержит наиболее общие свойства для данного PE-файла — тип процессора, для которого скомпилирован исполняемый файл; количество заголовков секций; размер опционального заголовка, а также некоторую другую информацию.

После файлового заголовка идет опциональный заголовок — IMAGE_OPTIONAL_HEADER32. Он содержит специфические

параметры данного PE-файла — размер исполняемого кода; относительный виртуальный адрес (RVA — Relative Virtual Address) точки входа в программу; базовый адрес, начиная с которого в память будет отображен образ исполняемого файла и многое другое. Также здесь хранится немаловажная информация о границах выравнивания секций в памяти — *SectionAlignment* и границах выравнивания секций в файле на диске — *FileAlignment*. Наличие двух видов выравнивания обусловлено спецификой работы компьютера — чтение информации с жесткого диска производится секторами, а из памяти — страницами. То есть если файл занимает 1 байт, с диска будет все равно прочитано 512 (один сектор). Поэтому размер файла на диске дополняется нулями (выравнивается) до определенного значения (обычно 512). То же самое в памяти, только там выравнивание обычно происходит на одну страницу (1000h). Проще говоря, выравнивание — это округление до определенной константы, нужное для оптимизации работы компьютера.

Прежде чем перейти к дальнейшему описанию, нужно прояснить одну важную вещь — PE-файл состоит из секций. Одна секция может содержать код, данные, таблицу импорта или что-либо еще. Главная сложность заключается в том, что секция на диске не обязательно соответствует секции в памяти. Это происходит по нескольким причинам: отчасти из-за выравнивания, о котором было сказано выше, а еще потому, что существует такая вещь, как неинициализированные данные. К примеру, если в листинге написать: buffer rb 1000h, то размер файла на диске не увеличится ни на байт. Другое дело в памяти, здесь он будет больше ровно на 1000h. Информация о секциях хранится в специальной таблице (Object Table). Она идет сразу же после PE-заголовка и состоит из набора заголовков секций. Каждый заголовок описывает одну секцию файла: ее размер, размещение на диске и в памяти, параметры и т.д. Все заголовки секции имеют длину 40 байт и располагаются без выравнивания.

За всеми заголовками в файле следуют тела секций. В Microsoft исторически сложились такие устойчивые названия секций:.text,.bss,.data,.rdata,.edata,.idata,.rsrc,.debug и некоторые другие. Однако совсем не обязательно, чтобы секции имели такие названия. Например, компиляторы от фирмы Inprise (бывшая Borland) присваивают секциям имена вида CODE, DATA и т.п. Кроме того, программист может создавать дополнительные секции со своими названиями. Наличие точки в начале секции также необязательно. Секции расположены вплотную друг к другу без промежутков, но это совсем не значит, что в самих сек-

циях не бывает пустых мест. Каждая стандартная секция имеет определенное назначение, например:

- в.text расположен код программы;
- в.bss размещаются неинициализированные данные;
- в.data — инициализированные данные;
- в.edata — функции, экспортируемые файлом;
- в.idata — таблицы импортируемых функций;
- в.rsrc — ресурсы;
- в сегменте .rdata — данные только для чтения (строки, константы, информация отладочного каталога).

Но это назначение секций не строгое, например .text вполне может быть начинен обычными данными, а также экспортируемыми функциями. Некоторые секции имеют особый формат.

В самом конце PE-файла за секциями вполне могут размещаться дополнительные данные, например какая-нибудь отладочная информация, но это носит необязательный характер.

Техника внедрения кода в PE-файлы. Существуют следующие способы внедрения кода в PE-файлы:

а) размещение кода поверх оригинальной программы (также называемое затиранием);

б) размещение кода в свободном месте программы (интеграция);

в) дописывание кода в начало, середину или конец файла с сохранением оригинального содержимого;

г) размещение кода вне основного тела файла-носителя (например, в динамической библиотеке).

Поскольку, способ а) приводит к необратимой потере работоспособности исходной программы и реально применяется только в вирусах, здесь он не рассматривается. Все остальные алгоритмы внедрения полностью или частично обратимы.

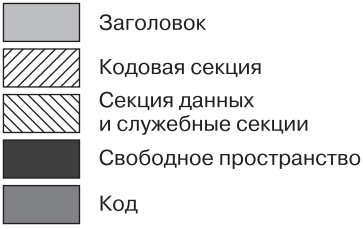


Рис. 4.6. Условные графические обозначения, принятые в дальнейшем изложении

Классификация механизмов внедрения. Механизмы внедрения можно классифицировать по-разному: по месту (начало, конец, середи-

на), «геополитике» (затирание исходных данных, внедрение в свободное пространство, переселение исходных данных на новое место обитания), надежности (предельно корректное, вполне корректное и крайне некорректное внедрение), рентабельности (рентабельное или нерентабельное) и т.д. Следующая классификация основана на характере воздействия на физический и виртуальный образ программы. Все существующие механизмы внедрения можно разделить на четыре категории.

К *категории A* относятся механизмы, не вызывающие изменения адресации ни физического, ни виртуального образов. После внедрения в файл ни его длина, ни количество выделенной при загрузке памяти не изменяется и все базовые структуры остаются на своих прежних адресах. Этому условию удовлетворяют: внедрение в пустое место файла (PE-заголовок, хвосты секций, регулярные последовательности), а также внедрение путем сжатия части секции.

К *категории B* относятся механизмы, вызывающие изменения адресации только физического образа. После внедрения в файл его длина увеличивается, однако количество выделенной при загрузке памяти не изменяется и все базовые структуры проецируются по тем же самым адресам, однако их физические смещения изменяются, что требует полной или частичной перестройки структур, привязывающихся к своим физическим адресам, и если хотя бы одна из них останется не скорректированной (или будет скорректирована неправильно), файл-носитель с высокой степенью вероятности откажет в работе. Категории B соответствуют: раздвижка заголовка, сброс части оригинального файла в оверлей и создание своего собственного оверлея.

К *категории C* относятся механизмы, вызывающие изменения адресации как физического, так и виртуального образов. Длина файла и выделяемая при загрузке память увеличиваются. Базовые структуры могут либо оставаться на своих местах (т.е. изменяются лишь смещения, отсчитываемые от конца образа (файла)), либо перемещаться по страничному имиджу произвольным образом, требуя обязательной коррекции. Этой категории соответствует: расширение последней секции файла, создание своей собственной секции и расширение серединных секций.

К «засекреченной» *категории Z* относятся механизмы, вообще не затрагивающие файла-носителя и внедряющиеся в его адресное пространство косвенным путем, например модификацией ключа реестра, ответственного за автоматическую загрузку динамических библиотек. Эту технологию в первую очередь используют сетевые черви и шпионы.

Категория А: *внедрение в пустое место файла*. Проще всего внедриться в пустое место файла. На сегодня таких мест известно три:

1) PE-заголовок;
2) хвостовые части секций;
3) регулярные последовательности.

Рассмотрим их подробнее.

Внедрение в PE-заголовок

Типичный PE-заголовок вместе с MS-DOS заголовком занимает порядка ~300h байт, а минимальная кратность выравнивания секций составляет 200h байт. Таким образом, между концом заголовка и началом первой секции практически всегда имеется ~100h байт, которые можно использовать для внедрения скрытого сообщения, размещая здесь либо всю внедряемую информацию целиком, либо ее часть, если размер не позволяет большего (рис. 4.7).

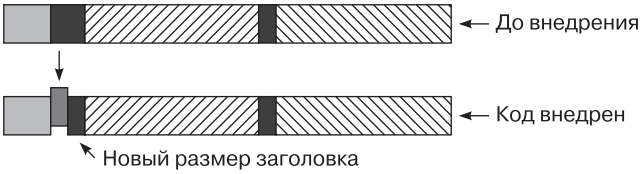


Рис. 4.7. Внедрение информации в свободное пространство хвоста PE-заголовка

Перед внедрением в заголовок необходимо убедиться, что хвостовая часть заголовка действительно свободна. Сканирование таких заголовков обычно выявляет длинную цепочку нулей, расположенных в его хвосте и, очевидно, никак и никем не используемых. Туда-то и можно записать информацию, но только с предосторожностями. Необходимо отсчитать по меньшей мере 10h байт от последнего ненулевого символа, оставляя этот участок нетронутым (в конце некоторых структур присутствует до 10h нулей, искажение которых ни к чему хорошему не приведет).

Внедрение в PE-заголовок в большинстве случаев можно распознать визуально. Рассмотрим, как выглядит в hex-редакторе типичный исполняемый файл notepad. exe (рис. 4.8): как уже было сказано выше, вслед за концом MS-DOS заголовка, обычно содержащим в себе строку This program cannot be run in DOS mode, расположена «PE» сигнатура, за которой следует немного мусора, разбавленного нулями и плавно перетекающего в таблицу секций, содержащую легко узнаваемые имена .text, .rsrc и .data.

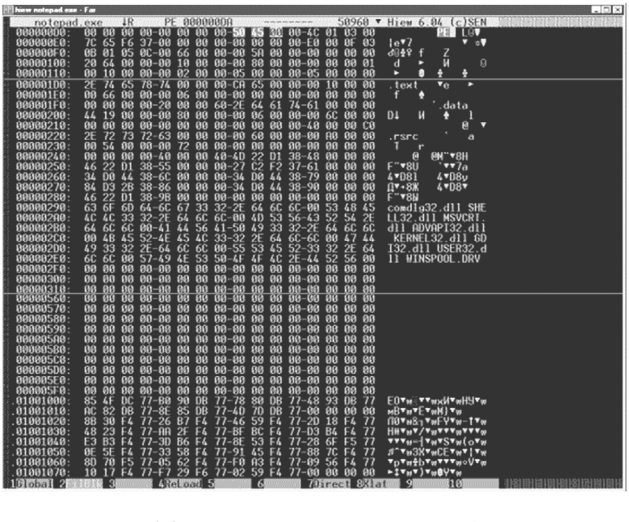


Рис. 4.8. Типичный PE-заголовок файла

Иногда за таблицей секций присутствует таблица BOUND импорта с перечнем имен загружаемых динамических библиотек. Дальше, вплоть до начала первой секции, не должно быть ничего, кроме нулей, использующихся для выравнивания. Если же это не так, то исследуемый файл содержит внедренный код.

На рисунке 4.8 показан типичный PE-заголовок файла, а на рис. 4.9 тот же файл после внедрения информации.

Рис. 4.9. Заголовок файла после внедрения

Внедрение в хвост секции

Операционная система Windows требует, чтобы физические адреса секций были выровнены по меньшей мере на 200h байт, поэтому между секциями практически всегда есть некоторое количество свободного пространства, в которое легко можно внедрить информацию (рис. 4.10).

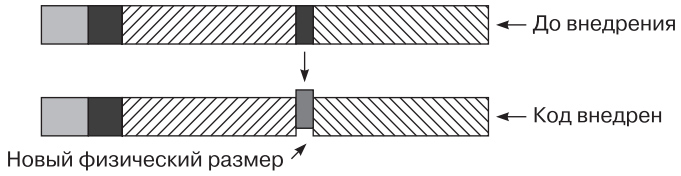


Рис. 4.10. Внедрение кода в хвост секции, оставшийся от выравнивания

Перед внедрением необходимо найти секцию с подходящими атрибутами и достаточным свободным пространством в конце или рассредоточить внедряемый код в нескольких секциях. Для этого необходимо просканировать хвостовую часть секции на предмет наличия непрерывной цепочки нулей — если таковая там действительно присутствует, ее можно безбоязненно использовать для внедрения.

Распознать внедрение этого типа достаточно трудно, особенно если код полностью помещается в первой кодовой секции файла, которой, как правило, является секция .text. Внедрение в секцию данных разоблачает себя наличием характерного кода в ее хвосте.

Внедрение осмысленного машинного кода в хвост секции данных представлено на рис. 4.11.

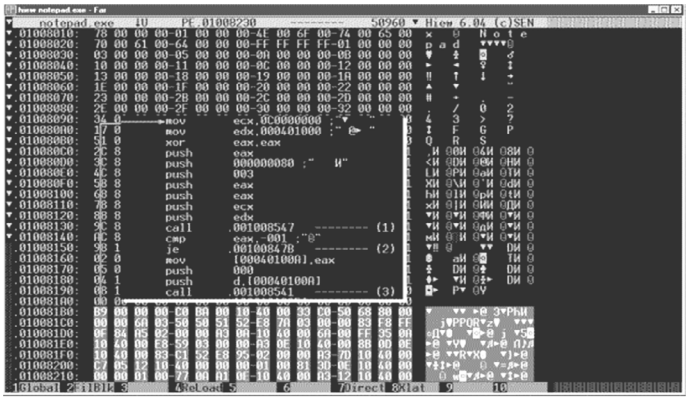


Рис 4.11. Осмысленный машинный код в хвосте секции данных

Внедрение в регулярную последовательность байт

Цепочки нулей необязательно искать в хвостах секций, также как необязательно искать именно нули, для внедрения подходит любая регулярная последовательность (например, цепочка FF FF FF.. или даже FF 00 FF 00...). Если внедряемых цепочек больше одной, код придется разбить по всему телу файла. Соответственно стартовые адреса и длины этих цепочек придется где-то хранить для их последующего восстановления.

Регулярные последовательности чаще всего обнаруживаются в ресурсах, а точнее — в bitmap'ах и иконках. Технически внедриться сюда ничего не стоит, но пользователь тут же заметит искажение иконки, чего допускать ни в коем случае нельзя (даже если это и не главная иконка приложения, так как проводник показывает остальные по нажатию кнопки «сменить значок» в меню свойств ярлыка). Внедрение кода в регулярные цепочки продемонстрировано на рис. 4.12.

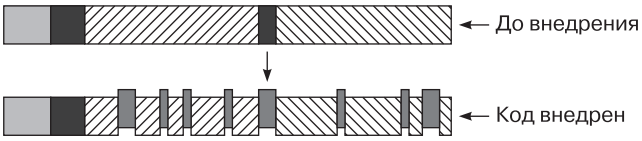


Рис. 4.12. Внедрение кода в регулярные цепочки

Алгоритм внедрения выглядит так: сканируем файл на предмет поиска регулярных последовательностей и отбираем среди них цепочки наибольшей длины, причем сумма их длин должна несколько превышать размеры кода, так как на каждую цепочку в среднем приходится 11 байт служебных данных: четыре байта на стартовую позицию, один байт — на длину, один — на оригинальное содержимое и еще пять байт на машинную команду перехода к другой цепочке.

Разбиваем код на части, добавляя в конец каждой из них команду перехода на начало следующей. Запоминаем начальные адреса, длины и исходное содержимое всех цепочек в импровизированном хранилище, сооруженном либо внутри PE-заголовка, либо внутри одной из цепочек. Если этого не сделать, потом будет невозможно извлечь код из файла. После чего записываем код в выбранные цепочки.

Категория A: *внедрение путем сжатия части файла*

Внедрение в регулярные последовательности фактически является разновидностью более общей техники внедрения в файл

путем сжатия его части, в данном случае осуществляемое по алгоритму RLE. Если же использовать более совершенные алгоритмы (например, Хаффмена или LZW), то стратегия выбора подходящих частей значительно упрощается. Мы можем сжать кодовую секцию, а на освободившееся место записать наш код. Для компрессии можно использовать функционал, реализованный в самой ОС (аудио, видео-кодеки, экспортеры графических форматов, сетевые функции сжатия и т.д.).

Естественно, упаковка оригинального содержимого секции (или ее части) не обходится без проблем. Во-первых, следует убедиться, что секция вообще поддается сжатию. Во-вторых, предотвратить сжатие ресурсов, таблиц экспорта (импорта) и другой служебной информации, которая может присутствовать в любой подходящей секции файла и кодовой секции в том числе. В-третьих, перестроить таблицу перемещаемых элементов (если, конечно, она вообще есть), исключая из нее элементы, принадлежащие сжимаемой секции и поручая настройку перемещаемых адресов непосредственно самому внедряемому коду (рис. 4.13).

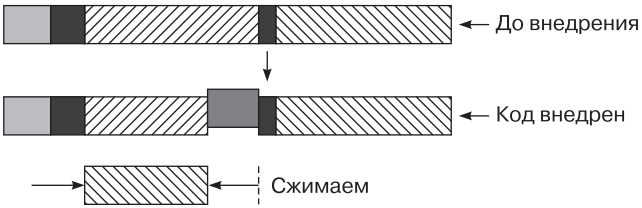


Рис. 4.13. Внедрение кода путем сжатия секции

Распознать факт внедрения в файл путем сжатия части секции трудно, но все-таки возможно. Дизассемблирование сжатой секции обнаруживает некоторое количество бессмысленного мусора, настораживающего опытного исследователя, но зачастую ускользающего от новичка. Также стоит обратить внимание на размеры секций. Если виртуальные размеры большинства секций много больше физических, то файл, по всей видимости, сжат каким-либо упаковщиком.

Категория B: *раздвижка заголовка*

Когда пространства, имеющегося в PE-заголовке (или какой либо другой части файла), оказывается недостаточно для размещения всего кода целиком, мы можем попробовать растянуть заголовок на величину, выбранную по своему усмотрению. До тех пор пока размер за-

головка (SizeOfHeaders) не превышает физического смещения первой секции, такая операция осуществляется элементарно, но вот дальше начинаются проблемы, для решения которых приходится кардинально перестраивать структуру исполняемого файла. Как минимум необходимо увеличить физические адреса начала всех секций на величину, кратную принятой степени выравнивания, прописанной в поле «Физическое выравнивание секций», и физически переместить хвост файла, записав код на освободившееся место.

Максимальный размер заголовка равен виртуальному адресу первой секции, так как заголовок не может перекрываться с содержимым страничного имиджа. Учитывая, что минимальный виртуальный адрес составляет 1000h, а типичный размер заголовка — 300h, мы получаем в свое распоряжение порядка 3 Кбайт свободного пространства, достаточного для размещения практически любого кода (рис. 4.14).

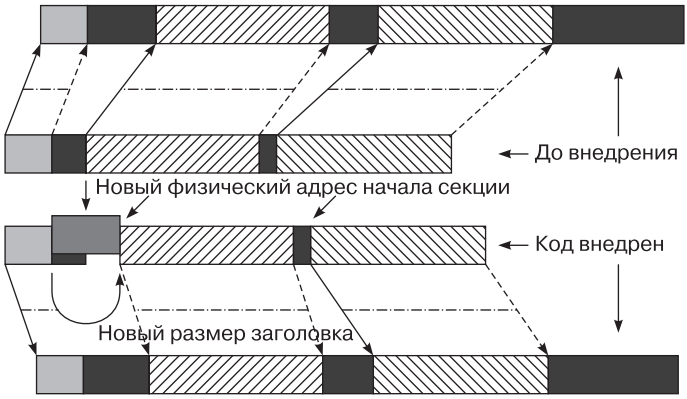


Рис. 4.14. Исполняемый файл и его проекция в память до и после внедрения кода путем раздвижки заголовка

Данный метод внедрения распознается аналогично обычному методу внедрения в PE-заголовок.

Описание программной реализации встраивания ЦВЗ. В программе ***Filigrana*** использован алгоритм, основанный на встраивании цифрового водяного знака (ЦВЗ) в свободное место исполняемого файла. Подробно данный алгоритм был рассмотрен в предыдущей главе (*Категория А*: *внедрение в пустое место*, *в конце секции файла*). В качестве ЦВЗ используется изображение формата BMP.

В общем виде алгоритм встраивания ЦВЗ представлен на рис. 4.15.

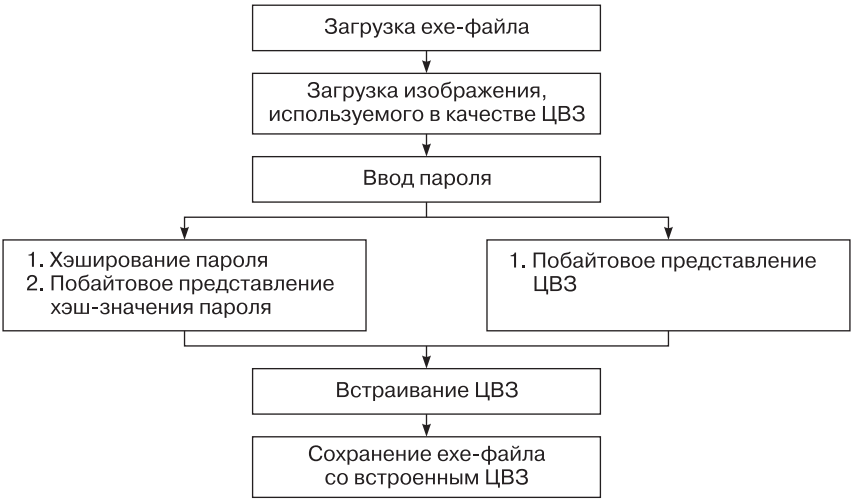


Рис. 4.15. Алгоритм встраивания ЦВЗ

Для демонстрации работы программы встраивания ЦВЗ возьмем два исполняемых файла:

1) программу Calc. exe из стандартной поставки Windows, размер программы на диске — 114 Кб (116 376 байт);

2) установочный файл SkypeSetup.exe программы Skype, размер файла на диске — 1,88 МВ (1 978 368 байт).

Встроим ЦВЗ наибольшего допустимого размера для каждого из исполняемых файлов.

В нашем случае для первого файла этот размер оказался равным 361 байт, о чем нам и сообщает программа (рис. 4.16).

Рис. 4.16. Сообщение о допустимом размере встраиваемого ЦВЗ

Для второго файла возможный размер ЦВЗ для встраивания увеличился до 13 668 байт (рис. 4.17).

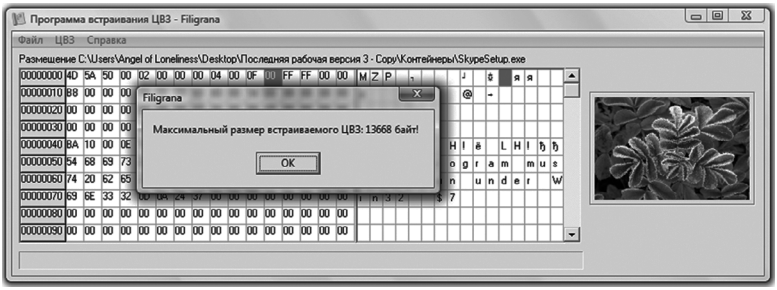


Рис. 4.17. Сообщение о допустимом размере встраиваемого ЦВЗ

При этом в процентном соотношении (размер встраиваемого ЦВЗ к размеру исполняемого файла) первый файл выигрывает у второго ~0,3% против ~0,15%. Но, как видно из выходных данных, наиболее эффективно использование программы для встраивания в файлы большего размера, что логично с точки зрения структуры исполняемых файлов, да и выявить факт встраивания в exe-файл большего размера будет сложнее из-за размеров дизассемблированого кода, при этом рекомендуется использовать ЦВЗ небольшого размера относительно размеров контейнера.

После встраивания размеры исполняемых файлов не изменились (рис. 4.18).

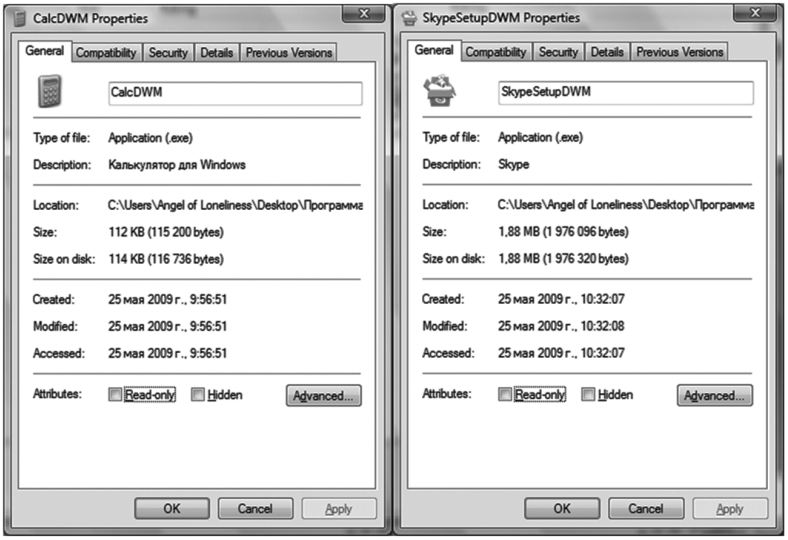
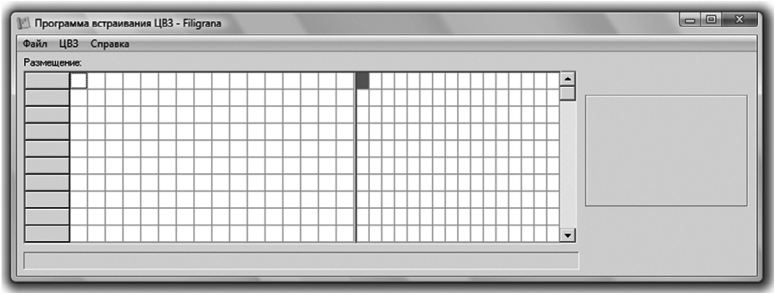


Рис. 4.18. Размеры exe-файлов, содержащих ЦВЗ

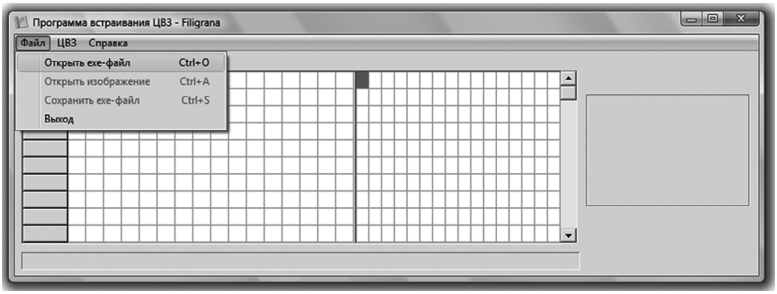
Задание

Встраивание цифрового водяного знака

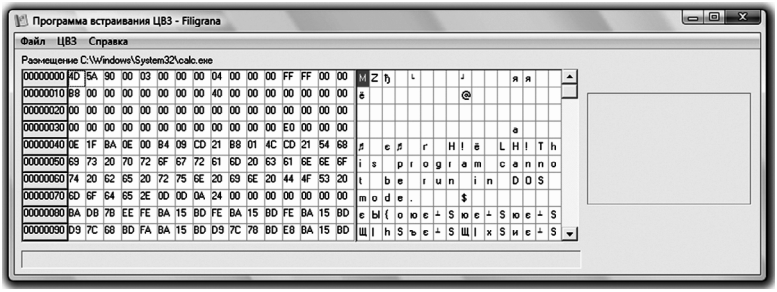
1. Запустите программу ***Filigrana***, предварительно установленную на ваш компьютер.



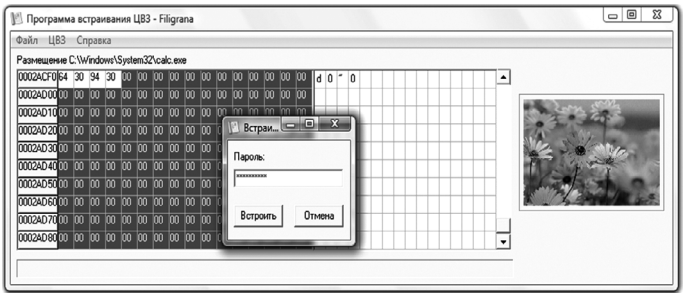
2. Выберите пункт «Открыть exe-файл» из меню «Файл» появившегося окна или нажмите комбинацию клавиш «Ctrl + O».



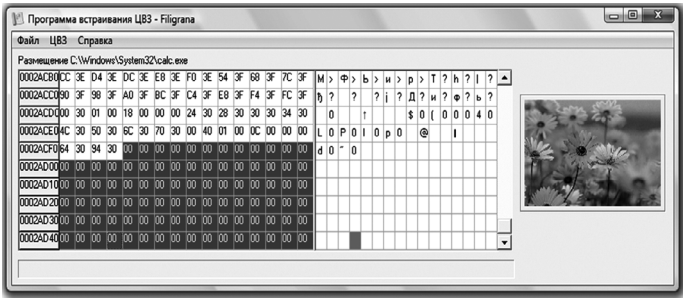
3. В появившемся диалоговом окне выберите необходимый exe-файл, два раза кликнув на нем левой кнопкой мыши или с помощью кнопки «Открыть». В окне главной формы отобразятся данные о выбранном файле в шестнадцатеричном и ASCII формате.



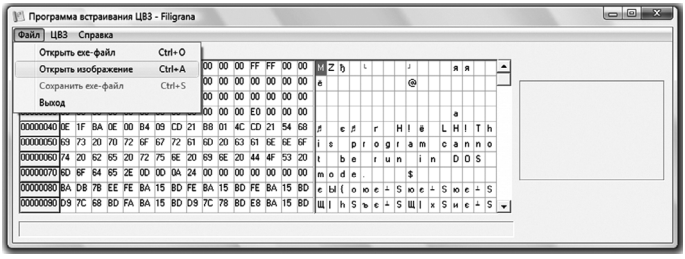
4. Выберите пункт «Открыть изображение» из меню «Файл» или нажмите комбинацию клавиш «Ctrl + A».



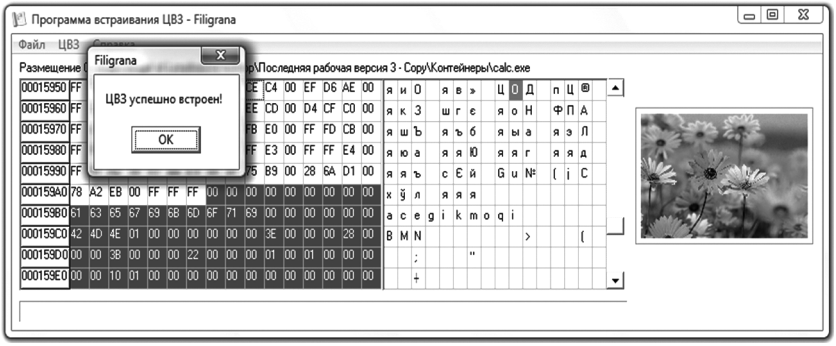
5. В появившемся диалоговом окне выберите изображение в формате .bmp, которое вы хотите использовать в качестве ЦВЗ, два раза кликнув на нем левой кнопкой мыши или с помощь кнопки «Открыть». В окне preview главной формы отобразится выбранное изображение, в hex-окне курсор автоматически переместится на строку, с которой начнется последующее встраивание цифрового водяного знака.



6. Чтобы встроить выбранное изображение, введите пароль (используйте сочетание букв, цифр и символов) в окне «Встраивание ЦВЗ», вызываемого из пункта «Встроить изображение» меню «ЦВЗ» главной формы или нажатием комбинации клавиш «Alt + I».



7. После ввода пароля нажмите кнопку «Встроить». В случае если выбранное изображение удовлетворяет условиям встраивания (размер ЦВЗ не превышает размер контейнера и exe-файл пригоден для встраивания), оно будет встроено, после чего появится соответствующее сообщение, а в hex-окне можно будет просмотреть результаты встраивания.

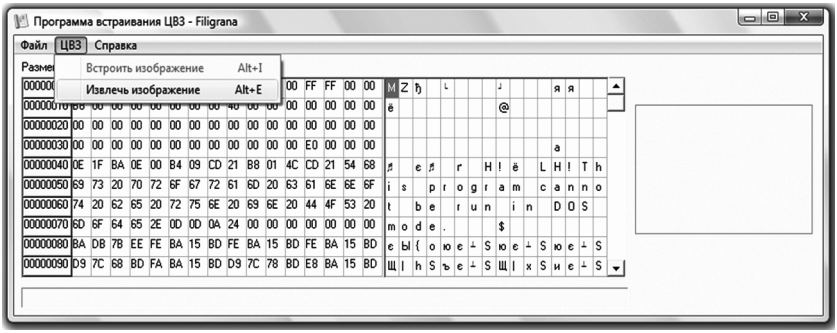


8. Для сохранения исполняемого файла со встроенным ЦВЗ выберите пункт «Сохранить exe-файл» из меню «Файл» или нажмите комбинацию «Ctrl + S», введите имя сохраняемого файла и нажмите кнопку «Сохранить».

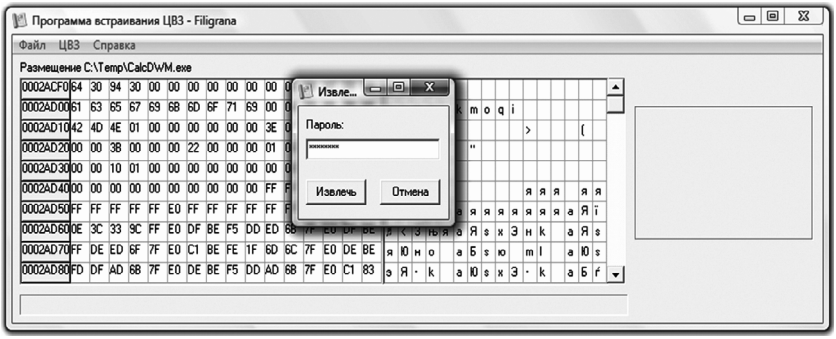
Внимание! Для успешного извлечения ЦВЗ необходимо запомнить свой пароль или сохранить его в месте, недоступном для других пользователей. Никому не показывайте свой пароль!

Извлечение цифрового водяного знака

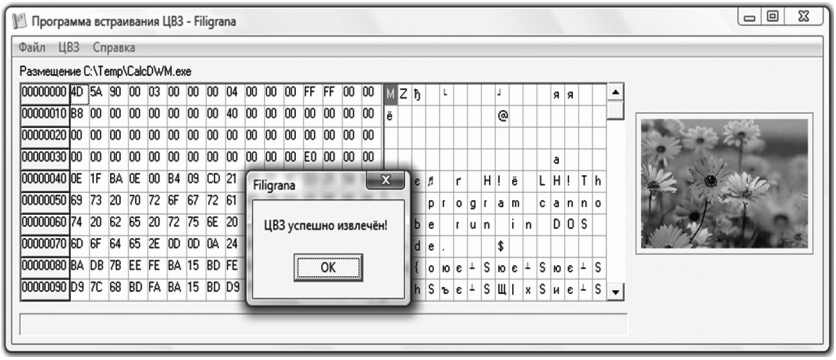
1. Откройте exe-файл, содержащий ЦВЗ. Выберите пункт «Извлечь изображение» меню «ЦВЗ» главной формы или нажмите комбинацию клавиш «Alt + E».



2. Для извлечения изображения в окне «Извлечение ЦВЗ» введите пароль, который использовался при встраивании цифрового водяного знака, и нажмите кнопку «Извлечь».



3. Если Вы ввели верный пароль, появится соответствующее сообщение, а в окне preview главной формы отобразится изображение, встроенное ранее.



4. Сохраните в отчете экранные формы, демонстрирующие процесс встраивания ЦВЗ в различные исполняемые файлы. Сделайте выводы об эффективности изученного метода защиты.

5. Включите в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 4.1.

Таблица 4.1

Номер варианта	Контрольные вопросы
1, 5, 7, 26	Перечислите способы защиты программных продуктов. Укажите их достоинства и недостатки
2, 4, 6	Какие методы включает юридическая защита программных продуктов? Охарактеризуйте основные из них
11, 13	Перечислите основные международные и отечественные источники защиты прав авторов программ
12, 14, 16	В чем заключается процедура лицензирования программ? Какими нормативными документами регулируется процесс лицензирования программ?
3, 9, 18, 29	Перечислите технические методы защиты программных продуктов. Кратко охарактеризуйте каждый из них
20, 22, 24	Какие методы стеганографии могут использоваться для защиты программных продуктов? Сравните методы стеганографической защиты и технической защиты ПО
10, 17, 19	Какие особенности структуры PE-файлов дают возможность эффективного внедрения цифровых водяных знаков?
21, 23, 25	Опишите суть метода внедрения кода в PE-файлы за счет размещения кода в свободном месте программы (интеграция)
8, 28, 27	Какой из методов внедрения кода в PE-файлы используется в программе *Filigrana*? Опишите суть этого метода, его достоинства и недостатки
12, 15, 30	Какие способы обнаружения, извлечения и модификации ЦВЗ вы можете предложить для изученного метода защиты ПО?

ЛАБОРАТОРНАЯ РАБОТА № 14

ЗАЩИТА ЭЛЕКТРОННЫХ ДОКУМЕНТОВ С ИСПОЛЬЗОВАНИЕМ ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ

Цель работы: ознакомление с методами защиты электронных документов с использованием цифровых водяных знаков.

Примечание. Для выполнения лабораторной работы на компьютере необходимо установить программы ***watermark*_1.05** и ***watermark*_2**.

Описание лабораторной работы. *Цифровые водяные знаки* (ЦВЗ) являются одной из наиболее перспективных областей использования компьютерной стеганографии. На электронные документы наносится специальная метка, которая остается невидимой для глаз, но распознается специальными программами. Такое программное обеспечение уже используется в электронных версиях некоторых журналов. Данное направление стеганографии призвано обеспечить защиту интеллектуальной собственности.

Водяные знаки могут быть как видимыми (или печатными), так и скрытыми (или цифровыми). Печатный водяной знак представляет собой некоторую видимую метку на изображении (рис. 4.19) — как правило, это подпись автора и (или) его логотип. Такие водяные знаки показывают, что данное изображение принадлежит конкретному автору или компании. Однако опытный компьютерный график при желании сможет удалить печатный водяной знак или заменить его на другой.

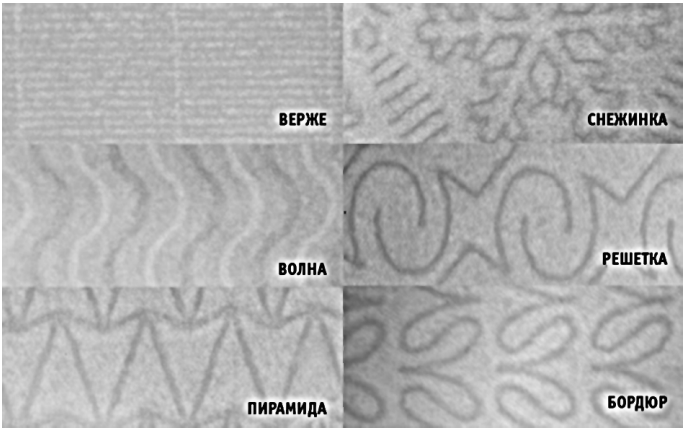


Рис. 4.19. Примеры печатных водяных знаков

Цифровой водяной знак — это цифровой код (как правило, он представляет собой идентификационные данные — ID, URL, адрес электронной почты, логотип и пр.), вставленный в электронный документ, чтобы идентифицировать информацию об авторских правах. В отличие от печатного водяного знака он невидим. ЦВЗ применяется для того, чтобы обеспечить защиту авторских прав для интеллектуальных ресурсов в цифровом формате. При этом биты информации, представляющие водяной знак, разбросаны внутри файла, поэтому они не могут быть идентифицированы или изменены.

ЦВЗ могут быть трех типов: *робастные*, *хрупкие и полухрупкие*. Под робастностью понимается устойчивость ЦВЗ к различного рода воздействиям на стего.

Хрупкие ЦВЗ разрушаются при незначительной модификации заполненного контейнера. Отличие от средств электронной цифровой подписи заключается в том, что хрупкие ЦВЗ все же допускают некоторую модификацию контента. Это важно для защиты мультимедийной информации, так как законный пользователь может, например, пожелать сжать изображение. Другое отличие заключается в том, что хрупкие ЦВЗ должны отразить не только факт модификации контейнера, но также вид и местоположение этого изменения.

Полухрупкие ЦВЗ устойчивы по отношению к одним воздействиям и неустойчивы по отношению к другим. Вообще говоря, все ЦВЗ могут быть отнесены к этому типу. Однако полухрупкие ЦВЗ специально проектируются так, чтобы быть неустойчивыми по отношению к определенного рода операциям. Например, они могут позволять выполнять сжатие изображения, но запрещать вырезку из него отдельных элементов или вставку в него фрагментов.

К цифровым водяным знакам предъявляются следующие требования:

- невидимость; когда на изображение ставится видимый логотип держателя прав, он, как правило, располагается в таком месте, где не сможет быть помехой для зрительного восприятия иллюстрации, следовательно и его умышленное удаление не повлечет за собой особых нарушений ее структурной целостности; исходя из сказанного, для предотвращения обнаружения и удаления метка должна быть невидима и хорошо скрыта данными исходного изображения, не допуская заметного искажения последнего не только из-за возможного обнаружения метки, но и из-за чисто эстетических соображений — не имеет смысла маркировать изображение, если его качество ухудшится настолько, что оно уже в принципе не будет похоже на оригинал;

- ЦВЗ должен быть устойчивым либо неустойчивым к преднамеренным и случайным воздействиям (в зависимости от приложения), если ЦВЗ используется для подтверждения подлинности, то недопустимое изменение контейнера должно приводить к разрушению ЦВЗ (хрупкий ЦВЗ), если же ЦВЗ содержит идентификационный код, логотип фирмы и т.п., то он должен сохраниться при максимальных искажениях контейнера, конечно, не приводящих к существенным искажениям исходного контента; например, у изображения могут быть отредактированы цветовая гамма или яркость, у аудиозаписи — усилено звучание низких тонов и т.д., кроме того, ЦВЗ должен быть робастным по отношению к аффинным преобразованиям изображения, т.е. его поворотам, масштабированию; при этом надо различать устойчивость самого ЦВЗ и способность декодера верно его обнаружить, скажем, при повороте изображения ЦВЗ не разрушится, а декодер может оказаться неспособным выделить его; существуют приложения, когда ЦВЗ должен быть устойчивым по отношению к одним преобразованиям и неустойчивым по отношению к другим, например, может быть разрешено копирование изображения (ксерокс, сканер), но наложен запрет на внесение в него каких-либо изменений;
- защищенность (стойкость к фальсификации); если метка должна быть устойчива к различного вида помехам и трансформациям, следовательно она должна быть устойчива к действиям, направленным на ее удаление;
- ЦВЗ должен иметь низкую вероятность ложного обнаружения скрытого сообщения в сигнале, его не содержащем, в некоторых приложениях такое обнаружение может привести к серьезным последствиям, например, ложное обнаружение ЦВЗ на DVD-диске может вызвать отказ от его воспроизведения плеером;
- ЦВЗ должен вычислительно легко извлекаться законным пользователем;
- должна существовать возможность добавления к стего дополнительных ЦВЗ, например, на DVD-диске имеется метка о допустимости однократного копирования, после осуществления такого копирования необходимо добавить метку о запрете дальнейшего копирования; можно было бы, конечно, удалить первый ЦВЗ и записать на его место второй, однако это противоречит предположению об устойчивости ЦВЗ к атакам удаления; лучшим выходом является добавление еще одного ЦВЗ, после которого первый не будет приниматься во внимание.

Удовлетворить всем этим требованиям не просто, однако существует множество компаний, предлагающих конкурирующие технологии и соответствующие программы для внедрения цифровых водяных знаков.

Основные области использования технологии ЦВЗ могут быть объединены в три группы: *защита от копирования*, *скрытая аннотация документов* и *доказательство аутентичности информации* (*аутентификация*) (рис. 4.20).

Рис. 4.20. Области применения ЦВЗ

Необходимо отметить, что наибольшие достижения стеганографии в прошедшем десятилетии были получены именно в области развития цифровых водяных знаков. Эти достижения вызваны реакцией общества на актуальнейшую проблему защиты авторских прав в условиях общедоступных компьютерных сетей.

Алгоритмы встраивания ЦВЗ в полутоновые и бинарные изображения. В настоящее время существует не так много алгоритмов, позволяющих встраивать цифровой водяной знак в полутоновые и бинарные изображения. Во многом это связано с тем, что разработчикам приходится выбирать между качеством изображения и робастностью встраиваемой информации. Далее будут рассмотрены основные принципы работы некоторых алгоритмов, встраивающих ЦВЗ в изображения, основными цветами которых являются градации серого.

Алгоритмы скрытия данных в частотной области изображения. Во многих методах скрытия данных в изображениях используется та или иная декомпозиция изображения-контейнера. Среди всех преобразований наибольшую популярность в стеганографии получило дискретное косинусное преобразование (ДКП), что отчасти объясняется его успешным применением при сжатии изображений

Обычно при встраивании информации с помощью ДКП контейнер разбивается на блоки 8×8 пикселей. Преобразование применяется к каждому блоку, в результате чего получаются матрицы коэффи-

циентов ДКП также размером 8×8 (рис. 4.21). Коэффициент в левом верхнем углу матрицы обычно называется DC-коэффициентом и содержит информацию о яркости всего блока. Остальные коэффициенты называются AC-коэффициентами. Основным недостатком данных алгоритмов является необходимость достаточно больших по размеру контейнеров.

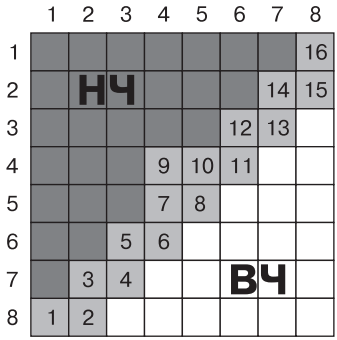


Рис. 4.21. Матрица ДКП

Алгоритм Хсу и Ву. Основной особенностью данного алгоритма является то, что декодеру ЦВЗ требуется исходное изображение. Однако декодер определяет не факт наличия ЦВЗ, а выделяет встроенные данные. В качестве ЦВЗ выступает черно-белое изображение размером вдвое меньше контейнера (рис. 4.22). Для большей стойкости и скрытности результатов использования данного стеганографического метода количество встраиваемой информации на практике пытаются уменьшить.

Рис. 4.22. Внедряемый ЦВЗ

Перед встраиванием изображение подвергается случайным перестановкам или перестановкам, зависящим от характеристик блоков контейнера, в который будет встроена информация (рис. 4.23). ЦВЗ

встраивается в среднечастотные коэффициенты ДКП (четвертая часть от общего количества). Эти коэффициенты расположены вдоль второй диагонали матрицы ДКП.

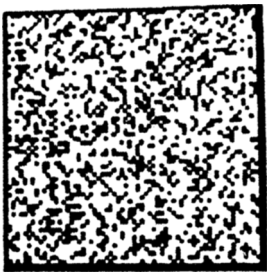


Рис. 4.23. Результат псевдослучайной перестановки элементов ЦВЗ

Для внедрения бита ЦВЗ s_i в коэффициент $c_b(j, k)$ находится знак разности коэффициента текущего блока и соответствующего ему коэффициента из предыдущего блока:

$$d_1(i) = \text{sign}(c_b(j, k) - c_{b-1}(j, k)).$$

Если необходимо встроить 1, коэффициент $c_b(j, k)$ меняют так, чтобы знак разности стал положительным, если 0 — то чтобы знак стал отрицательным.

Графическое представление извлеченного ЦВЗ, встраивание которого производилось путем изменения отношений между значениями ДКП соседних блоков, изображено на рис. 4.24.

Рис. 4.24. ЦВЗ, извлеченный из контейнера

Следует отметить, что этот алгоритм не является робастным по отношению к JPEG-компрессии, так же его отрицательными сторонами являются возможность встраивания ЦВЗ только в виде бинарного изображения и необходимость исходных данных для извлечения ЦВЗ.

Алгоритм Фридрих

Данный алгоритм является композицией двух алгоритмов: в одном данные встраиваются в низкочастотные, в другом — в среднечастотные коэффициенты ДКП. Каскадное применение двух различных алгоритмов приводит к хорошим результатам в отношении робастности. Это объясняется тем, что недостатки одного алгоритма компенсируются достоинствами другого. Исходный сигнал детектору ЦВЗ не требуется.

Перед встраиванием ЦВЗ в низкочастотные коэффициенты изображение преобразуется в сигнал с нулевым математическим ожиданием и определенным отклонением яркости так, чтобы абсолютные значения коэффициентов ДКП попали в определенный диапазон. Для этой цели используется следующее преобразование:

$$I \to \frac{1024}{\sqrt{XY}} \frac{I - \hat{I}}{\sigma(I)},$$

где X, Y — размеры изображения I в пикселях, $\hat{I}$ и $\sigma(I)$ — соответственно математическое ожидание и стандартное отклонение значений яркости пикселей. ЦВЗ представляет собой сигнал в виде последовательности чисел $\{-1; 1\}$.

Далее на основе геометрической прогрессии действительных чисел

$$t_0 = 1, \quad t_{i+1} = \frac{1+\alpha}{1-\alpha} t_i,$$

где параметр $\alpha \in (0, 1)$, строится функция

$$\text{ind}(t) = (-1)^i, \quad t > 1, \quad \tau \le t < \tau_{i+1},$$

позволяющая для каждого вещественного числа $t > 1$ определить его индекс. Этот индекс изменится только в том случае, если к числу t прибавить (отнять) число, превосходящее значение αt. На рисунке 4.25 показан вид функции $\text{ind}(t)$ для $\alpha = 0{,}1$, $\alpha = 0{,}2$, $\alpha = 0{,}3$.

Для внедрения бита ЦВЗ s_i в коэффициент c_j последний изменяется не более чем на 100α процентов так, чтобы $\text{ind}\left(\left|c_j'\right|\right) s_i$.

Операция извлечения проводится путем выполнения аналогичных с операцией встраивания преобразований контейнера, который подозревается на наличие скрытого сообщения.

В среднечастотные коэффициенты ДКП информация встраивается путем умножения преобразованного значения ЦВЗ на параметр α и сложения результата со значением коэффициента. Предварительное кодирование ЦВЗ выполняется по следующему алгоритму.

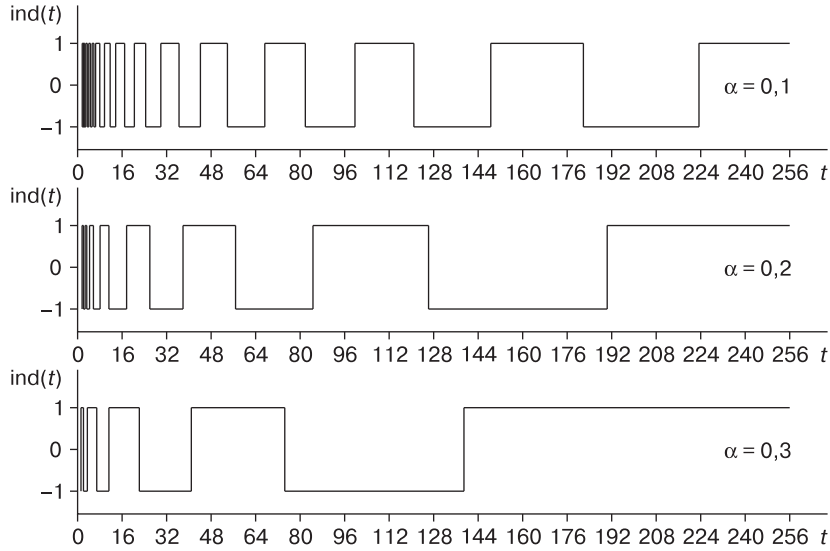


Рис. 4.25. Индексная функция ind(t)

Вход алгоритма: сообщение длины M, состоящее из символов $m_1 \in \{1, ..., B\}$.

Выход алгоритма: ЦВЗ длины N, состоящий из вещественных чисел s_i.

Для кодирования символа m_i генерируется $N + B + 1$ чисел псевдослучайной последовательности $r_i \in \{-1, 1\}$. Эту последовательность будем называть i-м случайным вектором.

Первые m_1 чисел этого вектора пропускаются, а следующие N чисел образуют вектор V_i, используемый при дальнейшем суммировании.

Для каждого символа сообщения генерируются статистически независимые различные случайные вектора.

В качестве ЦВЗ используется сумма векторов V_i.

В результате композиции встраивания в НЧ и СЧ-коэффициенты получился алгоритм, достаточно стойкий ко многим атакам. Такой алгоритм полезен для защиты произведений искусства и других полутоновых изображений, являющихся авторской собственностью.

Алгоритм В.А. Митекина. Рассматриваемый метод основан на кодировании встраиваемой информации количеством темных или светлых пикселей в блоке фиксированного размера. В данном случае используется кодирование одного бита встраиваемой информации битом четности блока 3×3 пикселя.

Предположим, что имеется бинарное изображение размера $M \times N$ пикселей. Разделим его на множество непересекающихся блоков B_i, каждый размера 3×3, где $1 \le i \le (M \times N) : (3 \times 3)$. В каждый из этих блоков будет внедрен бит информации s_i. Двумерный генератор псевдослучайных точек выбирает блок для внедрения. Решение о необходимости изменения бита в данном блоке принимается на основе формулы «контроля четности»:

$$H = \begin{cases} 0, s_i = \sum_{j=1}^{9} p_j \pmod 2 \\ 1, s_i \neq \sum_{j=1}^{9} p_j \pmod 2 \end{cases},$$

где p_j — значение функции яркости пикселя в рассматриваемом блоке B_i.

Если внедряемый бит соответствует биту четности H данного блока, то считается, что бит информации уже внедрен. Иначе необходимо инвертировать один пиксель в блоке. Выбор пикселя-кандидата для внедрения бита ЦВЗ основан на преобладающем цвете в блоке.

$$\text{Цвет кандидата} = \begin{cases} \text{Черный, если } N_B < 4 \\ \text{Белый, если } N_B \ge 7 \\ \text{Любой, если } 4 \le N_B < 7 \end{cases},$$

где N_B — количество темных пикселей в выбранном блоке.

Предположим, что существует L кандидатов в блоке, из которых нужно выбрать один. На окрестность каждого кандидата накладывается матрица веса размера 5×5, с помощью которой рассчитываем «вес» каждого кандидата.

Предположим, что кандидат имеет координаты (m, n). Тогда вес r рассчитывается следующим образом:

$$r(m,n) = \sum_{i=-2}^{2} \sum_{j=2}^{2} \left(\omega(i,j) \times I_{mn}(i,j) \right);$$

$$I_{mn}(i,j) = \begin{cases} 0, & \text{если } O(m,n) = O(m+i, n+j) \\ 1, & \text{если } O(m,n) \neq O(m+i, n+j) \end{cases},$$

где $O(m, n)$ — значение функции яркости пикселя исходного изображения с координатами (m, n);

$$W = \left(\omega_{i,j}\right) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 0 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \text{ — матрица веса.}$$

Вес $r(m, n)$ является мерой, характеризующей взаимосвязь кандидата и его соседних пикселей. Если найдено несколько кандидатов с одинаковым весом, то выбирается любой из них. В результате встраивания значение кандидата с наибольшим весом изменяется на противоположное. Таким образом, один блок содержит один бит ЦВЗ.

На рисунке 4.26 представлены увеличенный фрагмент оригинального изображения и фрагмент того же самого изображения после внедрения в него ЦВЗ объемом 40 бит.

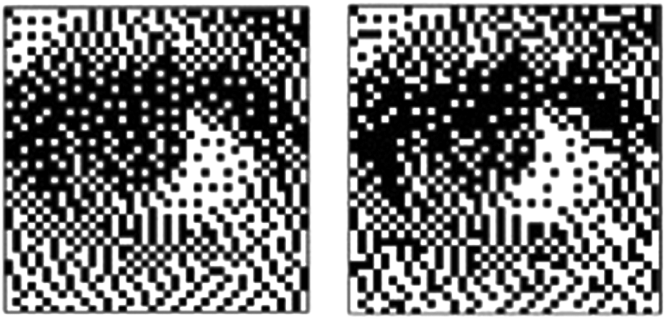


Рис. 4.26. Пример встраивания ЦВЗ по алгоритму Митекина

Рассмотренный метод кодирования является наиболее широко распространенным для встраивания ЦВЗ в бинарные изображения. На его основе разработан ряд модифицированных алгоритмов.

Одной из модификаций является расширение данного метода на полутоновые изображения. Изображение-контейнер представляет собой совокупность битовых «слоев», каждый из которых представляет изображение с глубиной цвета 1 бит. Таким образом, данный алгоритм применяется для одного или нескольких выбранных «слоев». Выбранный слой изображения модифицируется в соответствии с представленным выше алгоритмом. Результатом встраивания ЦВЗ является изображение той же глубины цвета, что и исходное.

Недостатком данного алгоритма является низкая степень сохранения визуального качества изображения. Во-первых, это связано с ма-

лыми размерами блоков для встраивания информации. Во-вторых, с выбором блоков для встраивания, ведь в зависимости от выбранного алгоритма встраивание может производиться и в полностью белые и в полностью черные блоки, что приведет к значительному искажению изображения-контейнера. В модификациях данного алгоритма предприняты попытки устранить данные недостатки.

Модификации алгоритма В.А. Митекина. Для программного обеспечения, встраивающего ЦВЗ в полутоновые изображения были разработаны две модификации алгоритма, описанного в работе В.А. Митекина. Обе модификации направлены на то, чтобы встроенная информация была незаметна в изображении-контейнере, т.е. на то, чтобы сделать алгоритм более робастным к атакам обнаружения.

Если встраивать биты информации во все подряд блоки изображения, то оно будет сильно искажено, а сам ЦВЗ будет легко обнаружен.

Было предложено следующее: в качестве ключа используется матрица K:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Алгоритм обработки, написанный на псевдокоде, выглядит следующим образом:

If (0<*sum* ($F_i \Lambda K$) <*sum* (K))
If (*sum* ($F_i \Lambda K$) *mod* 2 = b)
Keep F_i *intact*;
Else if (*sum* ($F_i \Lambda K$) *mod* 2 = 1)
Randomly pick a bit $[F_i]_{j,k} = 0$, $[K]_{j,k} = 1$ *and change* $[F_i]_{j,k}$ *to* 1;
Else if (*sum* ($F_i \Lambda K$) = *sum* (K) -1)
Randomly pick a bit $[F_i]_{j,k} = 1$, $[K]_{j,k} = 1$ *and change* $[F_i]_{j,k}$ *to* 0;
Else
Randomly pick a bit $[F_i]_{j,k}$, $[K]_{j,k} = 1$ *and complement* $[F_i]_{j,k}$;
Else
No data will be embedded in F_i *and* F_i *keeps intact*

На рисунке 4.27 изображен пример встраивания трех бит информации в черно-белое изображение размером 6×6. Изображение F делится на четыре равных блока размером 3×3, и в них по порядку встраивается информация.

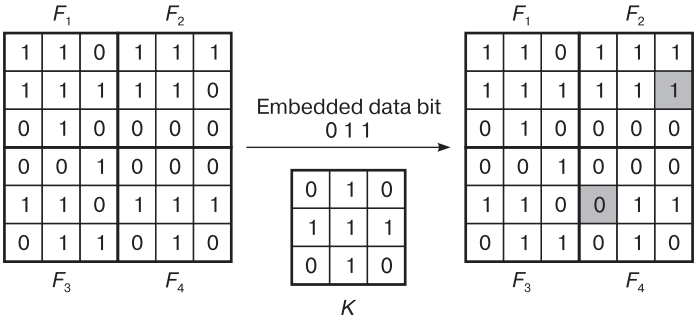


Рис. 4.27. Пример встраивания информации с использованием ключа.

F_1: sum($F_1 \Lambda K$) = 5 = sum(K) — в этот блок данные не встраиваются;

F_2: sum($F_2 \Lambda K$) = 3 — в этот блок можно встроить 1 бит данных, встраиваемый бит — 0, F'_2 — результат встраивания в соответствии с описанным выше алгоритмом;

F_3: sum($F_3 \Lambda K$) = 3 — в этот блок можно встроить 1 бит данных, встраиваемый бит — 1 = F_3 mod 2, блок контейнера изменять не нужно;

F_4: sum($F_4 \Lambda K$) = 3 — в этот блок можно встроить 1 бит данных, встраиваемый бит — 1, F'_4 — результат встраивания в соответствии с описанным выше алгоритмом.

Используя описанный выше алгоритм для определения, необходимы ли изменения в каком-либо конкретном блоке для встраивания бита информации, дальше действуем в соответствии с алгоритмом Митекина. Вместо случайного выбора бита для изменения выбираем наиболее подходящий бит для встраивания посредством наложения на кандидаты матрицы веса.

Недостатком этой модификации, как и исходного алгоритма, является маленький размер блоков, на которые делится изображение-контейнер. Кроме того, ключ, при помощи которого выбираются блоки для встраивания, позволяет избежать встраивания только в полностью белые блоки. Если изображение содержит крупные области черного цвета, то в заполненном контейнере на этом месте вследствие встраивания появятся строчки белых пикселей, что нанесет существенный ущерб как внешнему виду изображения, так и робастности ЦВЗ к атакам обнаружения.

С учетом перечисленных выше недостатков, был разработан еще один алгоритм, который является усложненной модификацией первых двух. Во-первых, в нем изображение уже разделяется не на бло-

ки 3×3, а на блоки 7×7 пикселей. Во-вторых, используется не один, а два ключа, что делает алгоритм выбора блоков сложнее, а самое главное, позволяет избежать встраивания как в полностью белые, так и в полностью черные блоки. Оба этих изменения в совокупности довольно значительно повышают робастность встраиваемого ЦВЗ к атакам обнаружения и степень сохранения визуального качества изображения.

Недостатком данного алгоритма является его ресурсозатратность. Так, для встраивания одного бита информации нам необходим блок размером 7×7. Таким образом, для встраивания одного символа необходимо восемь блоков или $49 \times 8 = 392$ пикселя. Кроме того, каждый ЦВЗ содержит символ-признак его начала и окончания. Получается, что для встраивания цифрового водяного знака, состоящего только из одного символа, требуется 24 блока 7×7, т.е. изображение, содержащее 1176 пикселей. И это — минимум, основанный на предположении о том, что все блоки будут пригодны для встраивания (пройдут проверку ключами).

Описание программного обеспечения для встраивания ЦВЗ. На основе модификаций алгоритма В.А. Митекина, описанных выше, предлагаются две программы с одинаковым интерфейсом, реализующие встраивание ЦВЗ двумя разными вариантами одного метода.

Описание структур и алгоритмов. Программы работают с файлами формата .bmp. Этот формат был выбран из-за его избыточности и возможности работать с ним в любой версии операционной системы (ОС) Windows за счет специальных встроенных функций. Программы позволяют встраивать цифровой водяной знак в монохромные изображения.

Обобщенный алгоритм работы программ представлен на рис. 4.28.

Описание интерфейса. Интерфейс обеих программ совершенно одинаков. Так же ничем не отличаются и основы работы с ними. Основные различия заключаются в принципах встраивания ЦВЗ, и, следовательно, как будет показано далее, в результатах встраивания.

Структура главного меню изображена на рис. 4.29.

Пункты меню File позволяют открывать изображения для встраивания (извлечения) ЦВЗ и сохранять модифицированные изображения. Пункты меню Edit запускают процессы встраивания и извлечения цифрового знака, Их дублируют кнопки Hide и Extract, расположенные под строкой ввода текста. В текстовой строке содержится либо текст встраиваемого ЦВЗ, набранный пользователем, либо текст извлеченной метки.

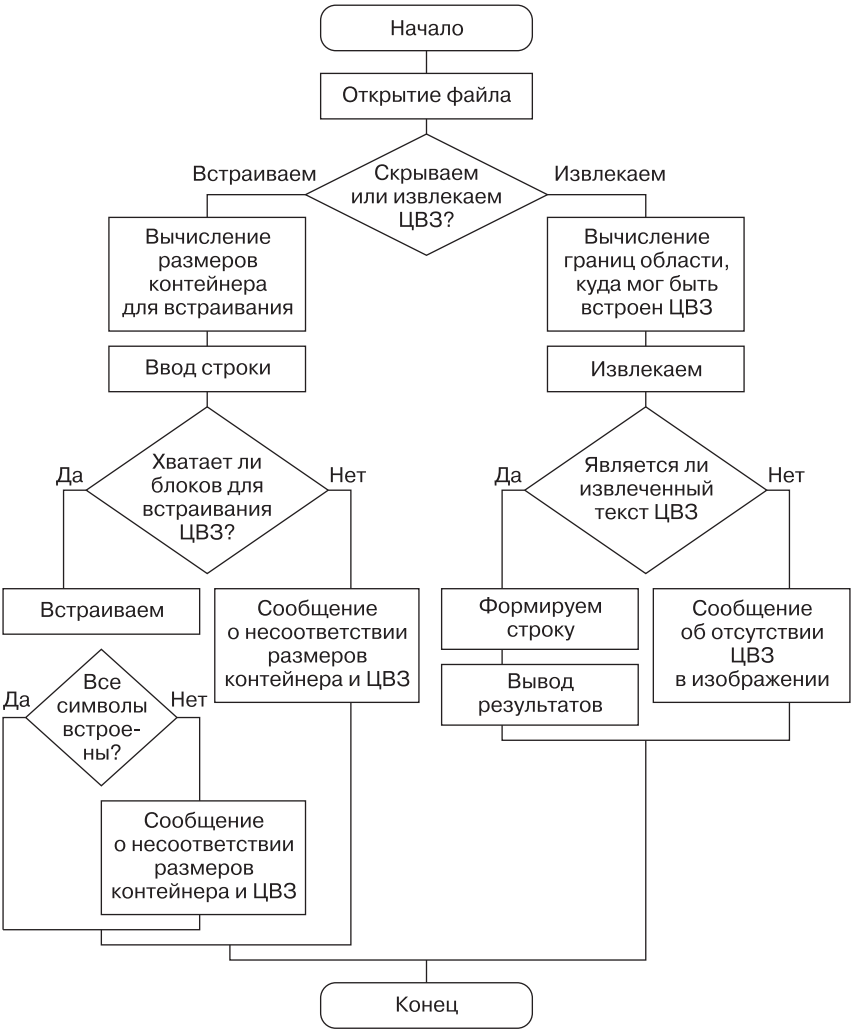


Рис. 4.28. Обобщенный алгоритм работы ПО, встраивающего ЦВЗ

Руководство по использованию ПО и примеры работы. Для работы с любой из программ прежде всего необходимо открыть подходящий файл .bmp. Для этого выбирается пункт меню File — Open и открывается диалоговое окно открытия файла.

Если выбранный файл представляет собой не черно-белое изображение, пользователю выдается сообщение об ошибке (рис. 4.30).

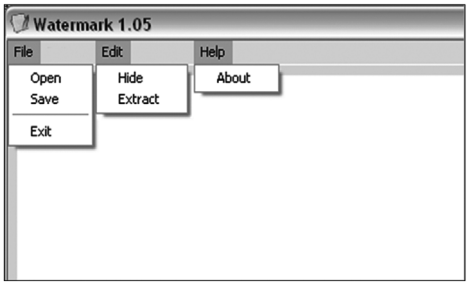


Рис. 4.29. Структура главного меню

Рис. 4.30. Сообщение о неподходящем формате открываемого файла

После открытия изображения пользователь выбирает — извлекать-ся будет ЦВЗ или встраиваться. Для встраивания в текстовую строку приложения должен быть введен текст метки (рис. 4.31).

Рис. 4.31. Цифровой водяной знак

После нажатия кнопки Hide или выбора соответствующего пункта меню начинается сам процесс встраивания. В случае если изображение-контейнер слишком мало для встраивания введенной в текстовой строке информации, выдается сообщение об ошибке (рис. 4.32). Если же процесс встраивания завершится успешно, модифицированное изображение будет выведено на экран и пользователь сможет его сохранить. Для этого выбирается пункт меню File — Save.

Рис. 4.32. Сообщение о том, что введенный ЦВЗ превышает допустимый для встраивания размер

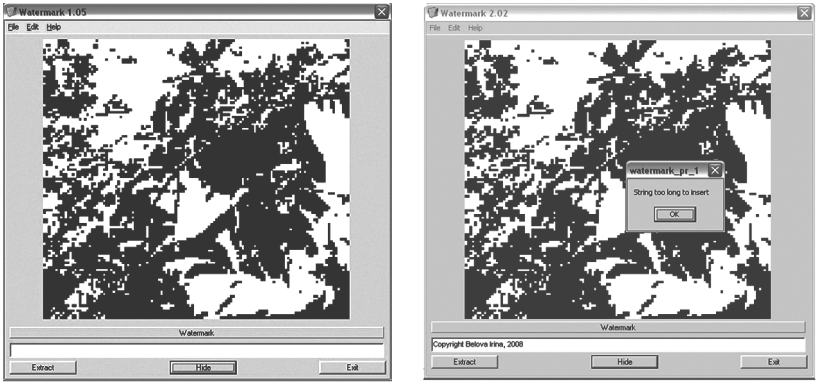
Для извлечения цифрового водяного знака необходимо нажать кнопку Extract или так же, как и при встраивании, выбрать соответствующий пункт меню. Если в изображении не будет обнаружено скрытой информации, пользователь получит об этом сообщение (рис. 4.33), иначе в текстовой строке будет выведен текст ЦВЗ (рис. 4.34).

Рис. 4.33. Сообщение о том, что ЦВЗ не обнаружен

Обе программы имеют свои преимущества и недостатки. Например, программа, встраивающая ЦВЗ в блоки изображения размера 3×3, имеет преимущества при работе с изображениями, размер которых не позволяет встроить большое количество символов. Для встраивания одного символа данной программе требуется всего 72 пикселя изображения, в то время как другой программе нужно 392 пикселя. Примеры встраивания ЦВЗ, состоящего из 28 символов (224 бита), обеими программами в изображение размерами 128×117 пикселей показаны рис. 4.35 и 4.36.

Рис. 4.34. Пример извлечения ЦВЗ

Рис. 4.35. Пустой контейнер размером 128×117 пикселей



а *б*

Рис. 4.36. Пример встраивания ЦВЗ в изображение размером 128×117 пикселей: а) в блоки 3×3 пикселя; б) в блоки 7×7 пикселей

При работе с изображениями, имеющими достаточно большие по размеру области черного цвета, лучше использовать программу, работающую с блоками изображения размером 7×7 пикселей. Программа, встраивающая метку в блоки 3×3 пикселя, в связи с наличием всего одного ключа показывает на таких изображениях недостаточно хорошие результаты по сохранению внешнего вида исходного изображения. Примеры работы программ для таких изображений представлены на рис. 4.37 и рис. 4.38. В изображение размером 400×300 пикселей встраивается ЦВЗ из 28 символов.

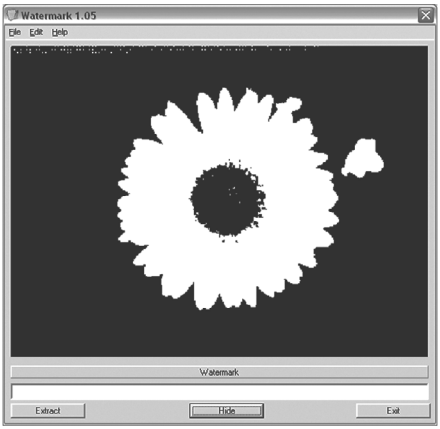


Рис. 4.37. Встраивание ЦВЗ в блоки изображения размером 3×3 пикселя

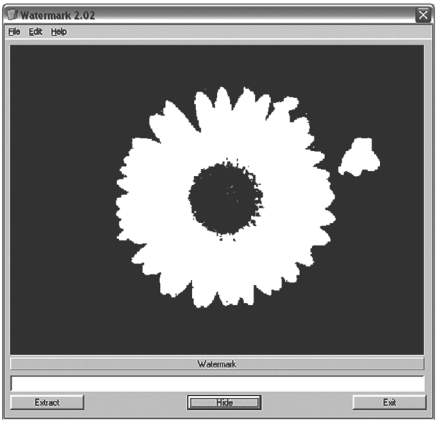


Рис. 4.38. Встраивание ЦВЗ в блоки изображения размером 7×7 пикселей

Задание

1. Запустите программу ***watermark*_1.05**, предварительно установленную на ваш компьютер.

2. Выберите пункт **Open** из меню **File** появившегося окна. Откройте, подготовленный заранее .bmp файл-контейнер с черно-белым изображением, в который необходимо встроить ЦВЗ (заранее необходимо подготовить три файла-контейнера: с большими областями белого цвета; с большими областями черного цвета; с чередующимися областями белого и черного цветов).

После открытия изображения необходимо выбирать — извлекаться будет ЦВЗ или встраиваться. Для встраивания в текстовую строку приложения должен быть введен текст метки (см. рис. 4.31). Подготовьте текст встраиваемого ЦВЗ и выполните встраивание. Сохраните файл со встроенной меткой в личной папке.

3. Действия, описанные в п. 2, повторите со всеми подготовленными файлами-контейнерами, не изменяя метку встраивания. Сравните полученные файлы-контейнеры на наличие визуальных искажений и по объему с исходными (непомеченными) файлами. Сделайте выводы об эффективности используемого алгоритма встраивания ЦВЗ.

4. Проведите извлечение ЦВЗ из всех помеченных файлов (процесс извлечения описан в разделе 3, см. рис. 4.32, 4.33). Сравните их качество, сделайте выводы об эффективности алгоритма извлечения ЦВЗ.

5. Помеченные файлы-контейнеры подвергните обработке с использованием:

— алгоритмов архивации с потерями;

— преобразований изображения (модификация, обрезание краев, масштабирование).

6. Повторите действия п. 4 по извлечению ЦВЗ из обработанных файлов-контейнеров. Сделайте выводы о робастности ЦВЗ.

7. Запустите программу ***watermark*_2.02**, предварительно установленную на ваш компьютер. Проделайте п. 2—5 с использованием программы ***watermark*_2.02**. Сравните результаты с полученными при использовании программы ***watermark*_1.05**. Сформулируйте выводы.

8. Сохраните в отчете экранные формы, демонстрирующие процесс встраивания ЦВЗ в различные файлы-контейнеры. Сделайте выводы об эффективности изученного метода защиты.

9. Включите в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 4.2.

Таблица 4.2

Номер варианта	Контрольные вопросы
1, 5, 7, 26	Перечислите способы защиты цифровых графических изображений от модификации и несанкционированного использования. Укажите их достоинства и недостатки
2, 4, 6	В чем особенность робастных, хрупких и полухрупких цифровых водяных знаков? Каковы ограничения в их использования для защиты электронных документов?
11, 13	Перечислите основные международные и отечественные источники защиты прав авторов цифрового графического контента
12, 14, 16	Алгоритм Хсу и Ву — особенности реализации, достоинства и недостатки
3, 9, 18, 29	Алгоритм Фридрих — особенности реализации, достоинства и недостатки
20, 22, 24	Какие методы стеганографии могут использоваться для защиты графических файлов? Сравните методы стеганографической защиты и криптографической защиты
10, 17, 19	Алгоритм В.А. Митекина — особенности реализации, достоинства и недостатки
21, 23, 25	В чем заключается усовершенствование алгоритма В.А. Митекина в программных продуктах, использованных в лабораторной работе? Оцените эффективность исследованного алгоритма для встраивания ЦВЗ?
8, 28, 27	Подсчитайте максимальных объем встраиваемого ЦВЗ в бинарное изображение размером 1024×128 пикселей: а) в блоки 3×3 пикселя, б) в блоки 7×7 пикселей
12, 15, 30	Подсчитайте максимальных объем встраиваемого ЦВЗ в полутоновое изображение (256 градаций серого) размером 512×256 пикселей: а) в блоки 3×3 пикселя, б) в блоки 7×7 пикселей

ЛАБОРАТОРНАЯ РАБОТА № 15

СТЕГОКОМПЛЕКСЫ, ДОПУСКАЮЩИЕ ИСПОЛЬЗОВАНИЕ АУДИОКОНТЕЙНЕРОВ, НА ПРИМЕРЕ ПРОГРАММЫ *INVISIBLE SECRETS*-4

Цель работы: изучение современных программных средств стеганографии на примере пакета *Invisible Secrets*-4.

Примечание. Для выполнения лабораторной работы на компьютере необходимо установить программу ***Invisible Secrets*-4**.

Описание лабораторной работы. Пакет *Invisible Secrets*-4 отличает проработанность пользовательского интерфейса, выполненного в виде «мастера», интегрированность в среду *Microsoft Windows* и удобство использования. Данный пакет программ может использовать в качестве контейнеров как аудио, так и графические файлы.

В качестве контейнеров данный программный продукт использует неупакованные аудиоданные, оцифрованные с разрядностью 8 или 16 бит на отсчет. Перед сокрытием сообщения оно шифруется. Доступны такие алгоритмы шифрования, как *Rijndael*, ГОСТ 28147—89, *Blowfish* и др.

При разработке стеганографических методов сокрытия информации основной целью является построение помехоустойчивого, необнаруживаемого метода. В связи с этим при рассмотрении программного продукта уделим особое внимание не интерфейсу, а реализованным методам.

Поскольку не для всех программных продуктов доступен исходный текст программы, то выводы часто приходится делать на основе анализа работы программы на тестовых контейнерах. В качестве тестовых контейнеров были выбраны: тональный сигнал с частотой 440 Гц и тишина (сигнал, у которого все отсчеты равны нулю). На рисунке 4.39 показан результат работы программы *Invisible Secrets*-4 на тестовых контейнерах при оцифровке 8 бит на отсчет.

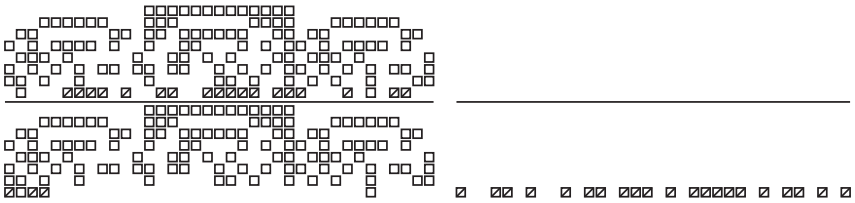


Рис. 4.39. Результат работы программы на тестовых контейнерах

На рисунке по горизонтали указываются отсчеты сигнала, по вертикали — разряды при записи отсчетов в двоичном виде. При наличии прямоугольника в соответствующем разряде соответствующего сэмпла стоит единица, в противном случае ноль. В верхней половине рисунка (выше разделительной линии) изображены отсчеты исходного сигнала контейнера, в нижней половине — отсчеты модифицированного сигнала контейнера. Перечеркнутые прямоугольники соответствуют несовпадающим битам в исходном и модифицированном контейнерах. На левом рисунке показан тональный сигнал, на правом — сигнал, содержащий тишину.

Программа *Invisible Secrets*-4 при внедрении данных в аудио контейнер использует наиболее распространенный в настоящее время метод модификации младшего бита (LSB-метод). Более того, при внедрении не производится никакого анализа содержимого контейнера, что видно из рисунка, где программа внедрила информацию в файл, содержащий тишину. Существенным недостатком данного программного продукта также является то, что при внедрении небольшого количества информации вся информация внедряется в начало контейнера, а не распределяется равномерно по всему контейнеру. Таким образом, возможно обнаружение наличия скрытого сообщения на основе анализа локальных различий статистик в разных частях контейнера.

На рисунке 4.40 приведен вид главного меню программы *Invisible Secrets*-4.

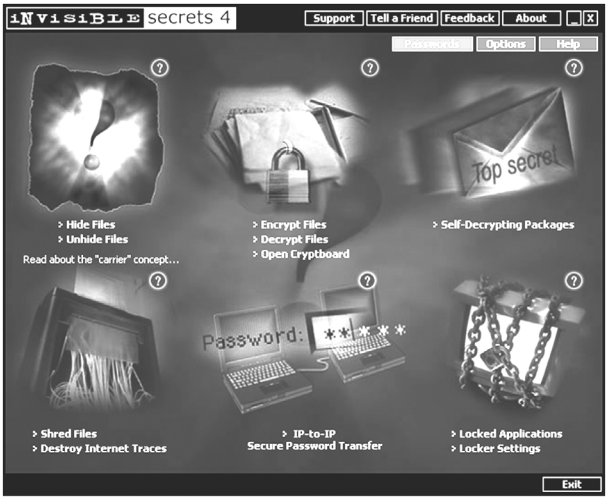


Рис. 4.40. Главное меню программы *Invisible Secrets*-4

Программа *Invisible Secrets*-4 состоит из шести функциональных модулей. Рассмотрим каждый из них отдельно.

Стеганография

Программа позволяет зашифровать и спрятать данные в таких местах, которые на поверхности выглядят совершенно безобидными, например в картинку, файлы с музыкой или веб-страницы. Такие типы контейнеров — прекрасная маскировка для секретной информации.

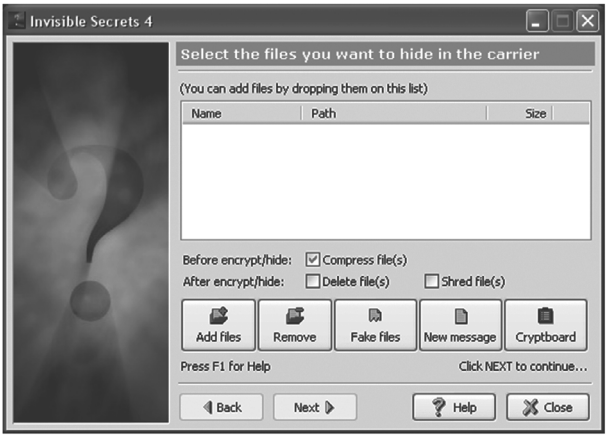


Рис. 4.41. Выбор файла, который необходимо встроить или зашифровать, и задание соответствующих параметров

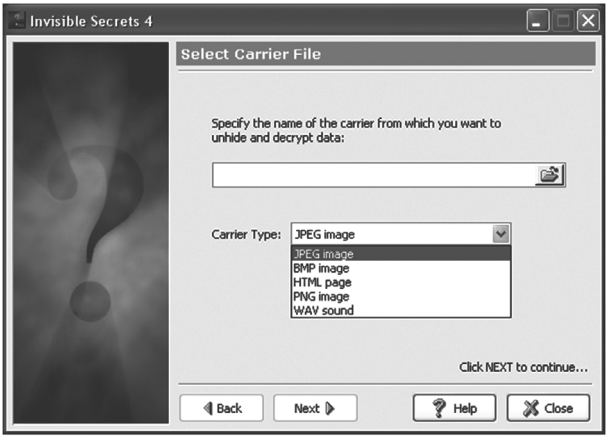


Рис. 4.42. Выбор файла для извлечения из файла-контейнера или расшифрования

Шифрование

Шифрование. В программе *Invisible Secrets*-4 предусмотрена защита файлов с использованием современных методов симметричного шифрования. Только владелец секретного пароля сможет прочесть их. Программа поддерживает такие алгоритмы шифрования, как: *Rijndael*, ГОСТ 28147—89, *Blowfish*, *Twofish*, *RC*4, *Cast*128, *Diamond* 2, *Sapphire II*. На рисунке 4.43 приведена панель шифрования файлов, а на рис. 4.44 панель расшифрования файлов.

Рис. 4.43. Панель шифрования файлов

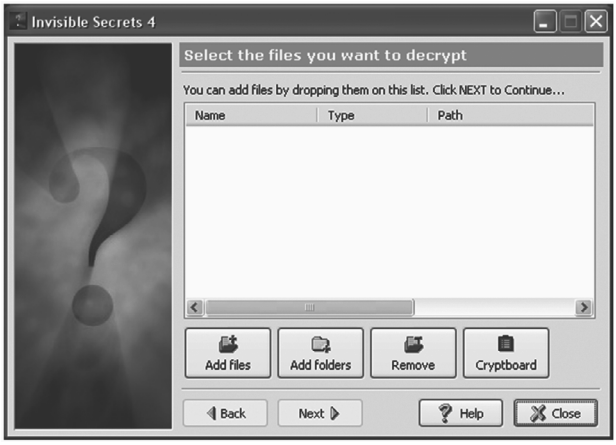


Рис. 4.44. Панель расшифрования файлов

Шифрование почтовых сообщений. Чрезвычайно актуальными являются процессы обмена конфиденциальной информацией в сети. При обмене личными сообщениями или коммерческими секретами возникает необходимость защищать конфиденциальную информацию от несанкционированного прочтения или модификации.. Программа *Invisible Secrets*-4 поможет отослать зашифрованное сообщение по *e-mail*, а получателю для его расшифрования необходимо будет знать только правильный пароль.

На рисунке 4.45 приводится панель программы *Invisible Secrets*-4 для создания самораспаковывающегося пакета для его отправки по *e-mail*.

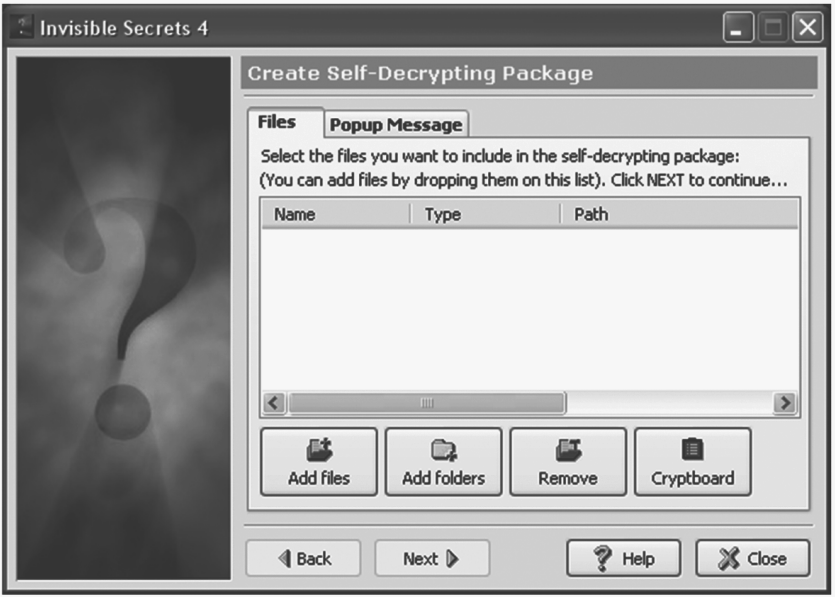


Рис. 4.45. Создание самораспаковывающегося пакета для его отправки по *e-mail*

Уничтожение папок, файлов и ссылок. Для уничтожения секретных папок или файлов без возможности их восстановления, а также различных ссылок в программе *Invisible Secrets*-4 используется модуль *DoD* 5220.22-*M Shredder*. Панель для уничтожения папок и файлов приведена на рис. 4.46, а для уничтожения ссылок на рис. 4.47.

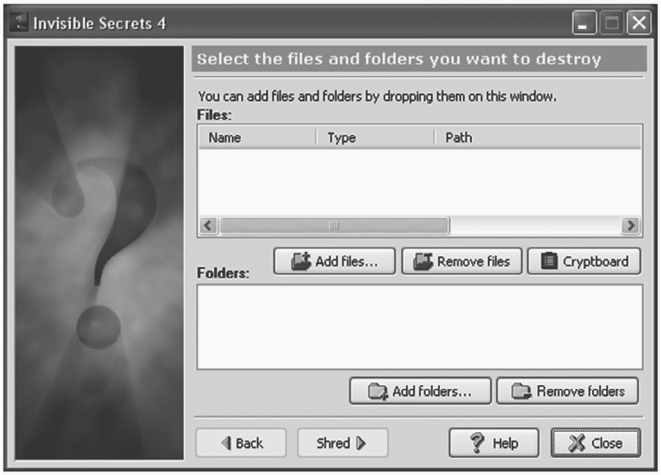


Рис. 4.46. Выбор папки или файла, которые необходимо уничтожить

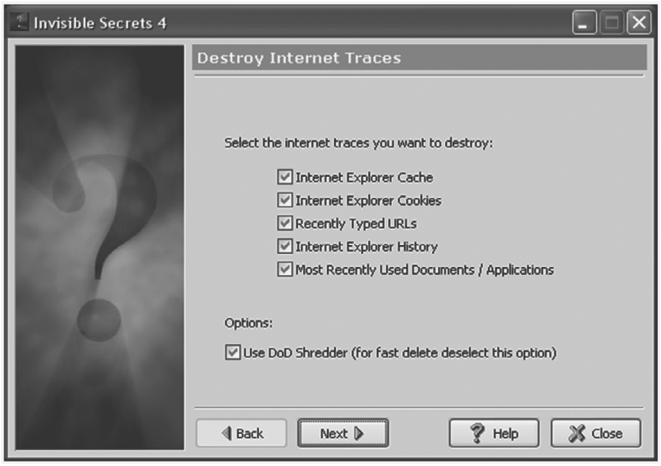


Рис. 4.47. Уничтожение ссылок

Передача паролей

Безопасная передача паролей. Одна из главных проблем, возникающих при организации защищенного обмена информацией, — это передача секретных паролей и ключей. *Invisible Secrets*-4 позволяет обмениваться паролями между двумя компьютерами при помощи создания безопасного соединения *IP*-к-*IP*.

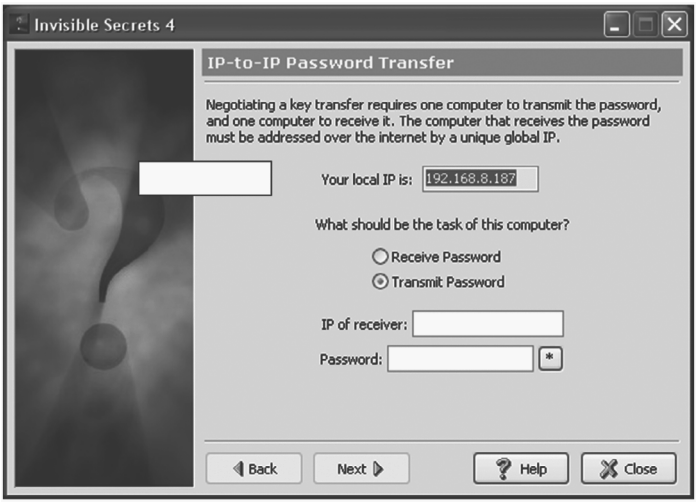


Рис. 4.48. Безопасная передача пароля

Задание

1. Установить на своем компьютере программу *Invisible Secrets*-4.
2. Ознакомиться со сведениями о программе *Invisible Secrets*, изложенными в разделе 1.
3. Запустить программу *Invisible Secrets*-4.
4. Изучить работу программы *Invisible Secrets* во всех шести возможных режимах:
   — стеганография;
   — шифрование;
   — шифрование почтовых сообщений;
   — уничтожение файлов;
   — передача паролей;
   — ограничение доступа к приложениям.
5. Сохранить в отчете экранные формы, демонстрирующие возможности программы *Invisible Secrets*-4 при работе в каждом из шести режимов.
6. Сделать выводы об эффективности используемого стеганографического пакета для каждого из рассмотренных режимов.
7. Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из табл. 4.3.

Таблица 4.3

Номер варианта	Контрольные вопросы
1, 5, 7, 3, 9, 18, 28	Сформулируйте основные отличия стеганографических и криптографических методов защиты информационных ресурсов. В чем достоинства и недостатки каждого из методов?
2, 4, 6, 8, 20, 22, 24, 26, 30	Перечислите шесть основных режимов работы программы *Invisible Secrets*-4. В каких случаях вы можете порекомендовать использование каждого из режимов?
11, 13, 15, 10, 17, 19, 27	Оцените эффективность работы программы *Invisible Secrets*-4 с графическими и аудиофайлами различных типов. Для этого в режиме «стеганография» встройте данные в контейнеры различных форматов, сравните пустые и заполненные контейнеры, сделайте выводы
12, 14, 16, 21, 23, 25, 29	При встраивании данных в режиме «стеганография» используйте различные алгоритмы для шифрования встраиваемых данных. Сравните пустые и заполненные контейнеры, как изменяется размер заполненных контейнеров в зависимости от метода шифрования? Почему вы получили такие результаты, обоснуйте

СПИСОК ЛИТЕРАТУРЫ

1. **Авдошин, С. М., Набебин, А. А.** Дискретная математика. Модулярная алгебра, криптография, кодирование. М.: ДМК Пресс, 2017. — 352 с.
2. **Бабаш, А. В., Шанкин, Г. П.** История криптографии. Ч. 1. Москва : Гелиос АРВ, 2002. — 240 с.
3. **Бабаш, А. В., Шанкин, Г. П.** Криптография / под редакцией В.П. Шерстюка, Э.А. Применко. Москва : СОЛОН-Р, 2002. — 512 с.
4. **Бабаш, А. В.** Криптографические и теоретико-автоматные аспекты современной защиты информации. Том 1. Москва : Изд. центр ЕАОИ, 2009. — 414 с.
5. **Бабаш, А. В., Баранова, Е. К.** Криптографические методы защиты информации: учебник. Москва : КноРус, 2016. — 190 с.
6. **Бабаш, А. В., Баранова, Е. К., Ларин, Д.** Информационная безопасность. История специальных методов криптографической деятельности. Москва : ИНФРА-М, РИОР, 2019. — 239 с.
7. **Баричев, С. Г.** Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. М.: СИНТЕГ, 2011. — 176 с.
8. **Герман, О. Н.** Теоретико-числовые методы в криптографии / О.Н. Герман, Ю.В. Нестеренко. М.: Академия, 2012. — 272 с.
9. **Грибунин, В. Г., Оков, И. Н., Туринцев, И. В.** Цифровая стеганография. Москва : СОЛОН — Пресс, 2002. — 272 с.
10. **Гурин, А. В.** Технологии встраивания цифровых водяных знаков в аудиосигнал / А.В. Гурин, А.А. Жарких, В.Ю. Пластунов. М.: Горячая линия — Телеком, 2015. — 116 с.
11. **Зубов, А. Ю.** Криптографические методы защиты информации. Москва : Гелиос АРВ, 2005. — 192 с.
12. **Зубов, А. Н.** Математика кодов аутентификации / А.Н. Зубов. М.: Гелиос АРВ, 2007. — 288 с.
13. **Конахович, Г. Ф., Пузыренко, А.Ю.** Компьютерная стеганография. Теория и практика. Москва : МК — Пресс, 2006. — 286 с.
14. **Коутинхо, С.** Введение в теорию чисел. Алгоритм RSA. М.: Постмаркет, 2001. — 328 с.
15. **Рябко, Б. Я.** Криптографические методы защиты информации: Учебное пособие для вузов / Б.Я. Рябко, А.Н. Фионов. М.: Горячая линия — Телеком, 2014. — 229 с.
16. **Смарт, Н.** Криптография. Москва : Техносфера, 2006. — 528 с.

17. **Черемушкин, А. В.** Лекции по арифметическим алгоритмам в криптографии. Москва : МЦНМО, 2002. — 104 с.
18. **Шеннон, К.** Работы по теории информации и кибернетике. М.: ИЛ, 1963. — 830 с. (Раздел — Теория связи в секретных системах.)
19. **Шнайер, Б.** Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си: моногр. / Брюс Шнайер. М.: Триумф, 2012. — 816 с.

ТЕСТОВЫЕ ЗАДАНИЯ

1. Всякий источник сообщений можно моделировать списком допустимых (т.е. встречающихся в каких-либо текстах) k-грамм при k=1,2,3,… Какие из приведенных k-грамм не являются допустимыми в русском языке? (несколько верных ответов)

1) “ШЕЕ”;
2) “ЖЫФ”;
3) “АУ”;
4) “ЮЪХ”;
5) “ЖЬН”.

2. Криптограмма получена в результате простой замены:
«ВГАДЮБКГЖЯАО МЮБ ЕДБЕБЗ ЛЖФАЮТ АЬЯБГЫЖРАА»
Ключ-подстановка:

А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Ж	З	Х	К	И	Ц	Ч	Л	А	В	Ъ	Ы	Ь	Б	Д	Г	Е	Ю	Э	Я	П	Р	У	С	Ф	Ш	Т	Щ	М	Н	О

Восстановленный исходный текст:

1) «КРИПТОЛОГИЯ ЭТО НАУКА О ЗАЩИТЕ ИНФОРМАЦИИ»;

2) «КРИПТОГРАФИЯ ЭТО СПОСОБ ЗАЩИТЫ ИНФОРМАЦИИ»;

3) «КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ».

3. Что означает термин «многократное шифрование» применительно к блочным шифрам?

1) повторное применение алгоритма шифрования к зашифрованному тексту с теми же ключами;

2) шифрование одного и того же блока открытого текста несколько раз с несколькими ключами;

3) увеличение числа этапов шифрования открытого текста.

4. Гаммирование чаще всего осуществляется: (несколько верных ответов)

1) по модулю 2, если открытый текст представляется в виде бинарной последовательности;

2) по модулю 256, если открытый текст представляется в виде последовательности байтов;

3) по модулю 16, если открытый текст представлен в цифровом виде;

4) по модулю 10, если открытый текст представлен в виде последовательности цифр, что иногда делается в ручных системах шифрования.

5. Основой построения большинства поточных шифров являются:

1) генераторы псевдослучайных чисел, в частности, различные комбинации регистров сдвига;

2) схемы суммирования по mod 16;

3) таблицы подстановок.

6. Зашифрованный методом перестановки открытый текст: «Сертификаты ключей ЭЦП», при ключе длинной 7, и перестановке: {4132756}, имеет вид:

1) тСреиифыктал кйюечЦ Э П ;

2) юклчТи ЭСЦ еиртфаикт ы ;

3) чКилют рСекиафиЭтПы Ц .

7. Зашифровать слово «выборочность» методом перестановки с ключом {3142}:

1) бвоычрнотоьс ;

2) ыовбрчоонсьт ;

3) ывброончотсь .

8. Зашифровать открытый текст – «field» методом Виженера, ключ – «toon» (алфавит – латиница):

1) rwcup

2) rwsyp

3) rvsyp

9. Частотный анализ может эффективно применяться для дешифрования шифров:

1) перестановки;

2) многоалфавитной замены;

3) простой замены.

10. Какие меры практической стойкости шифра относительно метода криптоанализа вы можете выделить: (несколько верных ответов)

1) вероятность дешифрования за время, не превосходящее T;

2) среднее время, необходимое для дешифрования шифра;

3) скорость дешифрования шифра.

11. Какие шифры можно называть имитостойкими?

1) шифры, обладающие свойством противостоять разрастанию ошибок при расшифровании текстов;

2) шифры, обладающие свойством противостоять попыткам навязывания ложной информации.

12. Какие шифры можно называть помехоустойчивыми?

1) шифры, обладающие свойством противостоять разрастанию ошибок при расшифровании текстов;

2) шифры, обладающие свойством противостоять попыткам навязывания ложной информации.

13. Разрастание числа ошибок означает что

1) ошибка в одной букве, допущенная при шифровании, приводит к большому числу ошибок в расшифрованном тексте;

2) ошибка в одной букве, допущенная при расшифровании, приводит к последующим ошибкам.

14. Шифр считается совершенным,

1) если он не поддается дешифрованию;

2) если положение противника, стремящегося к его дешифрованию, не облегчается в результате перехвата зашифрованного текста;

3) если требуются большие затраты или мала вероятность успеха его дешифрования.

15. Шифр считается практически стойким,

1) если он не поддается дешифрованию;

2) если положение противника, стремящегося к его дешифрованию, не облегчается в результате перехвата зашифрованного текста;

3) если требуются большие затраты или мала вероятность успеха его дешифрования.

16. Степень неоднозначности восстановления открытого текста при дешифровании

1) возрастает при уменьшении материала;

2) снижается при уменьшении материала.

17. Какова длина ключа в отечественном стандарте симметричного шифрования:

1) 56 бит;

2) 124 бит;

3) 256 бит

18. Что позволяет предотвратить использование криптографических систем с открытым ключом?

1) отказ от информации;

2) передачу секретного ключа получателю;

3) передачу открытого ключа получателю;

4) использование алгоритмов асимметричного шифрования.

19. Ниже перечислены механизмы защиты АИС от несанкционированного доступа и взлома. Что здесь лишнее?

1) идентификация и аутентификация пользователей и субъектов доступа;
2) управление доступом;
3) обеспечение постоянного числа пользователей сети;
4) обеспечения целостности.

20. Какова длина блока алгоритма шифрования DES?

1) 16 бит;
2) 56 бит;
3) 64 бита;
4) 5 байт.

21. Сколько всего циклов выполняется операция зашифровывания в алгоритме шифрования DES?

1) 10;
2) 14;
3) 16;
4) 20.

22. Какой алгоритм не используется при симметричном шифровании?

1) поточное шифрование;
2) шифрование методом гаммирования;
3) блочное шифрование;
4) алгоритм Эль-Гамаля.

23. Что является преимуществом симметричного шифрования?

1) скорость выполнения криптографических преобразований;
2) легкость внесения изменений в алгоритм шифрования;
3) секретный ключ известен только получателю информации и первоначальный обмен не требует передачи секретного ключа;
4) применение в системах аутентификации (электронная цифровая подпись).

24. Длина раундового ключа в отечественном стандарте симметричного шифрования:

1) 8 бит;
2) 32 бита;
3) 48 бит.

25. Как иначе называется асимметричное шифрование?

1) шифрование с закрытым ключом;
2) шифрование методом Бейтса;

3) шифрование с открытым ключом;
4) шифрование с переменным ключом.

26. Что является преимуществом асимметричного шифрования?
1) скорость выполнения криптографических преобразований;
2) легкость внесения изменений в алгоритм шифрования;
3) секретный ключ известен только получателю информации и первоначальный обмен не требует передачи секретного ключа;
4) применение в системах аутентификации (электронная цифровая подпись).

27. В асимметричных алгоритмах шифрования ключи зашифровывания и расшифровывания всегда:
1) разные, хотя и связанные между собой;
2) разные, никак не связанные между собой;
3) совпадают;
4) ключ зашифровывания представляет собой ключ расшифровывания, записанный в обратном порядке.

28. Безопасность системы RSA основана на:
1) трудности задачи разложения на простые множители;
2) комбинации символов, выбранных случайным образом;
3) использовании секретного ключа для шифрования;
4) использовании простого делителя в качестве открытого ключа.

29. Что лежит в основе криптографического контроля целостности:
1) правильная полная архивация;
2) хэш-функция;
3) кодовая защита;
4) биометрическая идентификация.

30. Хэш-функция используется для (несколько верных ответов):
1) для создания сжатого образа сообщения, применяемого в ЭЦП;
2) для быстрой передачи данных;
3) для идентификации отправителя;
4) для защиты пароля;
5) для построения кода аутентификации сообщений.

31. Какую роль выполняет электронная цифровая подпись
1) информация о передаваемых данных;
2) роль обычной подписи в электронных документах;
4) дополнительная информация;
5) обратный адрес.

32. При формировании цифровой подписи по классической схеме отправитель: (несколько правильных ответов)

1) применяет к исходному тексту хэш-функцию;

2) применяет к исходному тексту идентификатор отправителя;

3) дополняет хэш-образ до длины, требуемой в алгоритме создания ЭЦП;

4) выполняет максимальное сжатие;

5) вычисляет ЭЦП по хэш-образу с использованием секретного ключа создания подписи.

Таблица ответов

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

Варианты заданий с решениями

Вариант 1

Расшифровать криптограмму, полученную с использованием шифра простой замены, при имеющемся ключе шифрования.

Текст: ВГАДЮБКГЖЯАОМЮБЕДБЕБЗЛЖФАЮТАЬЯБГЫЖРАА

Ключ-подстановка:

А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Ж	З	Х	К	И	Ц	Ч	Л	А	В	Ъ	Ы	Ь	Б	Д	Г	Е	Ю	Э	Я	П	Р	У	С	Ф	Ш	Т	Щ	М	Н	О

Варианты ответов:

1. СПОСОБ ЗАЩИТЫ ИНФОРМАЦИИ ЭТО КРИПТОГРАФИЯ
2. КРИПТОГРАФИЯ БОЛЕЕ НАДЕЖНЫЙ СПОСОБ ЗАЩИТЫ
3. КРИПТОГРАФИЯ ЭТО СПОСОБ ЗАЩИТЫ ИНФОРМАЦИИ

Ответ: 3

Решение:

Запишем **ключ-обратной подстановки** на основе ключа-подстановки:

А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
И	О	К	Р	П	С	А	Б	Д	Г	З	Э	Ю	Я	Х	Ц	Ш	Ы	Ч	Щ	В	Е	Ж	Ъ	Ь	Л	М	Н	У	Т	Ф

При помощи этого ключа расшифровываем текст. В соответствии с ключом-обратной подстановки первая буква зашифрованного текста «В» перейдет в «К», «Г» перейдет в «Р» и так далее. В итоге получим расшифрованный текст:

КРИПТОГРАФИЯ ЭТО СПОСОБ ЗАЩИТЫ ИНФОРМАЦИИ

Вариант 2

Расшифровать криптограмму: 13 34 22 24 44 34 15 42 22 34 43 45 32.

Известно, что криптограмма получена с использованием квадрата Полибия:

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

Варианты ответов:

1. ERGO SUM COGITO
2. COGITO ERGO SUM
3. COGITO SUM ERGO

Ответ: 2

Решение:

Нумеруем строки и столбцы квадрата Полибия:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Буквы исходного текста находятся на пересечении соответствующей строки и столбца квадрата, например, 13 (строка 1, столбец 3 – буква исходного текста «С») и т.д.

В итоге, получаем расшифрованный текст: COGITO ERGO SUM (лат. «Я мыслю, следовательно, существую», Р. Декарт).

Вариант 3

Дана таблица замен:

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е
2	Е	Ж	З	И	Й	К
3	Л	М	Н	О	П	Р
4	С	Т	У	Ф	Х	Ц
5	Ч	Ш	Щ	Ъ	Ы	Ь
6	Э	Ю	Я			

Известно, что буква «О» в исходном тексте, заменена числом 34 в криптограмме.

Какая из криптограмм соответствует исходному тексту: КРИПТОГРАФИЯ?

Варианты ответов:

1. 26 36 24 35 42 34 41 24 41 42 16 63
2. 26 36 24 35 42 34 14 36 11 32 32 11
3. 26 36 24 35 42 34 14 36 11 44 24 63

Ответ: 3

Вариант 4

Зашифрован исходный текст: FIELD методом Виженера, ключ (лозунг): KEY (алфавит – латиница 25 символов, исключена буква «W»). Какая из криптограмм верная?

Варианты ответов:

1. PXCVH
2. PMCVH
3. RMCYH

Ответ: 2

Решение:

Исходный текст: F I E L D

Лозунг (с повторением по длине исходного текста): K EY KE

Буква исходного текста – строка; буква лозунга – столбец. На пересечении строки-F и столбца-K находим букву зашифрованного текста: «P», и так далее.

Правильный ответ:
PMCVH

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

Вариант 5

Расшифровать криптограмму, полученную с помощью метода случайного гаммирования, считая, что буквы алфавита пронумерованы от 0 до 32, соответственно. Зная определенную гамму.

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	

Текст: ХСЩРАБЛЮТЕЧЙ

Гамма(ключ):

11	1	17	1	14	19	9	14	19	17	15	11

Варианты ответов:
1. КРИПТОГРАФИЯ
2. КРИПТОЗАЩИТА
3. КАРДИОГРАФИЯ

Ответ: 1

Решение:

Y	X	C	Щ	P	A	Б	Л	Ю	T	E	Ч	Й
	22	18	26	17	0	1	12	31	19	5	24	10
К	11	1	17	1	14	19	9	14	19	17	15	11
X	11	17	9	16	19	15	3	17	0	21	9	32
	К	Р	И	П	Т	О	Г	Р	А	Ф	И	Я

Первая строка - зашифрованный текст.

Вторая строка – номера букв зашифрованного текста.

Третья строка – гамма.

Четвертая строка – номера букв открытого текста в алфавите.

Пятая строка – открытый текст, в соответствии с номерами.

Вычитаем из номера буквы алфавита соответствующий компонент ключа, если разность меньше 0, то прибавляем 33. Например первая буква (22-11=11), а пятая буква (0-14=-14; -14+33=19). Затем записываем в расшифрованный текст в соответствии с номером в алфавите: КРИПТОГРАФИЯ

Вариант 6

Зашифровать фразу: СООБЩИТЕ О ПРИБЫТИИ ЛИНЕЙНЫХ КОРАБЛЕЙ шифром вертикальной перестановки (фраза для шифрования записывается слитно без пробелов). В качестве секретного ключа (лозунга) используется фраза: БОЖЕ ЦАРЯ ХРАНИ, фраза записывается слитно, и буквы нумеруются в алфавитном порядке (при этом, если буква встречается несколько раз, номера ей присваиваются последовательно):

Б	О	Ж	Е	Ц	А	Р	Я	Х	Р	А	Н	И
3	8	5	4	12	1	9	13	11	10	2	7	6

Варианты ответов:
1. ИЕИРЫСЫЕБИБОИАБКИХОТРТНЙПНОЙЩЛЛЕЕ
2. ЕЕИРЫСЫОБИБОИАБКИХОТРТНЙПНОЙЩЛЛИИ
3. ИИЕРЫСЫОБИБОИАБКИХОТРТНЙПНОЙЩЛЛЕЕ

Ответ: 3

Решение:

Ключевая последовательность является так называемым номерным рядом. Записываем исходный текст без пробелов: СООБЩИТЕОПРИБЫТИИЛИНЕЙНЫХКОРАБЛЕЙ. При шифровании выписывается номерной ряд, а под ним шифруемый текст:

3	8	5	4	12	1	9	13	11	10	2	7	6
С	О	О	Б	Щ	И	Т	Е	О	П	Р	И	Б
Ы	Т	И	И	Л	И	Н	Е	Й	Н	Ы	Х	К
О	Р	А	Б	Л	Е	Й						

Шифрование производится выпиской по колонкам номерного ряда. В первой колонке стоят буквы ИИЕ, во второй РЫ, в третьей СЫО и т.д.

Получим шифрованный текст:

ИИЕРЫСЫОБИБОИАБКИХОТРТНЙПНОЙЩЛЛЕЕ

Вариант 7

Зашифровать открытый текст: ШИФРОВАНИЕ_ПЕРЕСТАНОВКОЙ, используя метод усложненной перестановки. Запись исходного текста по строкам некоторой матрицы, ключ k1 {5-3-1-2-4-6}, чтение зашифрованного текста по столбцам этой матрицы ключ k2 {4-2-3-1}.

Варианты ответов:

1. ПСНОРЙЕРВАИК_ЕАНФОИЕОТШВ
2. ПСНОРЙЕРВА_ИКЕАНФОИЕОТШВ
3. ЕАНФОИЕОТШВ_ПСНОРЙЕРВАИК

Ответ: 1

Решение:

1	И	Е	–	П
2	Е	Р	Е	С
3	О	В	А	Н
4	Т	А	Н	О
5	Ш	И	Ф	Р
6	В	К	О	Й
k_1/k_2	1	2	3	4

Открытый текст: «ШИФРОВАНИЕ_ПЕРЕСТАНОВКОЙ».

Матрица из четырех столбцов. Ключи: k_1 {5-3-1-2-4-6}; k_2 {4-2-3-1}.

Запись по строкам производится в соответствии с ключом k_1.

Чтение по столбцам в соответствии с ключом k_2.
Шифротекст: «ПСНОРЙЕРВАИК_ЕАНФОИЕОТШВ».

Вариант 8

Расшифровать криптограмму, полученную с использованием шифра перестановки, имея ключ.

Криптограмма: ПОРИКТФЧРАГИА СКЕЯИААЩЗТ

Ключ:

1	2	3	4	5	6
5	3	4	1	6	2

Варианты ответов:

1. КРИПТОЗАЩИТА ИНФОРМАЦИИ
2. КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА
3. ЗАЩИТА КРИПТОГРАФИЧЕСКАЯ

Ответ: 2

Решение:

Составим **ключ-обратной подстановки**:

1	2	3	4	5	6
4	6	2	3	1	5

Разобьём текст в соответствии с длиной ключа и запишем в таблицу:

П	О	Р	И	К	Т
Ф	Ч	Р	А	Г	И
А		С	К	Е	Я
И	А	А	Щ	З	Т

Переставим столбцы в соответствии с ключом-обратной подстановки. Первый столбец станет четвертым, второй – шестым, и так далее. После перестановки получим таблицу:

К	Р	И	П	Т	О
Г	Р	А	Ф	И	Ч
Е	С	К	А	Я	
З	А	Щ	И	Т	А

Затем выпишем строки по порядку и получим расшифрованный текст:

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА

Баранова Елена Константиновна
Бабаш Александр Владимирович

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ
Лабораторный практикум

Учебное пособие

Редактор *Н.А. Смирнова*
Корректор *Н.С. Орлова*
Верстка *С.Ю. Родионовой*