**КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ И СРЕДСТВА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**Часть 3**

# Содержание

# ЛАБОРАТОРНАЯ РАБОТА №9

# ШИФР ПЛЕЙФЕРА

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

## 1.Теоретические сведения

### Шифр Плейфера

Шифр Плейфера или квадрат Плейфера – ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 году 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. Он может быть проведен, но не для 26 возможных символов (латинский алфавит), а для 26×26=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 матрицы 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.

Для того 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. В некоторых вариантах шифра Плейфера вместо «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 преобразуются в символы того же столбца, находящееся 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 заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

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

**ПРИМЕР**

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

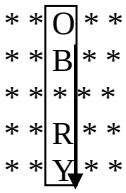
1. Биграмма HI формирует прямоугольник, заменяем её на BM.
2. Биграмма DE расположена в одном столбце, заменяем её на ND.
3. Биграмма TH формирует прямоугольник, заменяем её на ZB.
4. Биграмма EG формирует прямоугольник, заменяем её на XD.
5. Биграмма OL формирует прямоугольник, заменяем её на KY.
6. Биграмма DI формирует прямоугольник, заменяем её на BE.
7. Биграмма NT формирует прямоугольник, заменяем её на JV.
8. Биграмма HE формирует прямоугольник, заменяем её на DM.
9. Биграмма TR формирует прямоугольник, заменяем её на UI.
10. Биграмма EX находится в одной строке, заменяем её на XM.
11. Биграмма ES формирует прямоугольник, заменяем её на MN.
12. Биграмма TU находится в одной строке, заменяем её на UV.
13. Биграмма MP формирует прямоугольник, заменяем её 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»

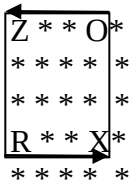
**ПРИМЕР**

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

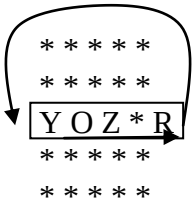

**OR** заменяется на **YZ**




**OR** заменяется на **BY**




**OR** заменяется на **ZX**




**OR** заменяется на **ZY**

## 2. Описание программы ШИФР ПЛЕЙФЕРА (The Playfair cipher)

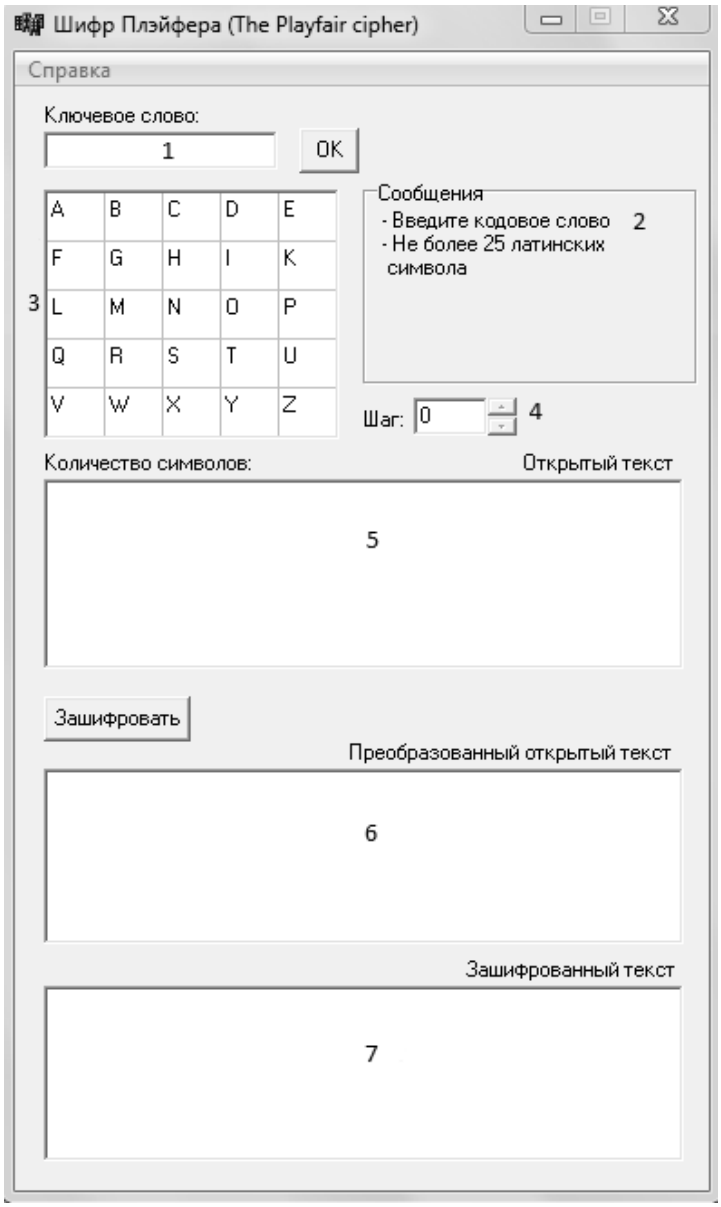



Рис. 9.1. Главное окно программы

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 ввода исходного текста.
6. Поле для вывода преобразованного открытого текста.
7. Поле для вывода зашифрованного текста.

## 2. Порядок выполнения лабораторной работы

Для выполнения лабораторной работы на компьютере необходимо установить программу *Playfair.exe*. Запустить 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 вопросы, выбранные в соответствии с номером варианта из *приложения 1*.

*Приложение 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 с полученными 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.

## 1.Теоретические сведения

### 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 её двумя произвольными буквами (например, Ь, Э) до ближайшего числа, кратного 5, то есть 25. Выпишем эту дополненную фразу без пропусков, одновременно разбив её на группы по пять символов, в соответствии со степенью подстановки:

СВЯЩЕ ННАЯР ИМСКА ЯИМПЕ РИЯЬ

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

ЩВСЕЯЯННРАКМИАСПИЯЕМЬИРЭЯ

При расшифровании текст разбивается на группы по 5 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 $1\times2\times3\times\ldots\times n$ *(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 не 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:

$$H= P_i Log_2(1/P_i)$$

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, то есть 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 ему, 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 века 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:

$$H= P_i( P_{ij}\times Log_2(1/P_{ij}))$$

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 по программированию. 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 так далее. 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 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(АЗ)+ k(Е )+ k(ГТ)+ k(ИП)=7+9+0+5=21*
*k (1-3) =k(АЮ)+ k(ЕС)+ k(ГО)+ k(ИЕ)=6+8+8+8=30*
*k (1-4) =k(АЖ)+ k(ЕШ)+ k(ГО)+ k(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:*

О Е Р Т
Я И И
О Ф Р Н
Ц А И М
О П Р И

*Зашифрованный текст*: ОЕРТ ЯИИОФРНЦАИМОПРИ

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:

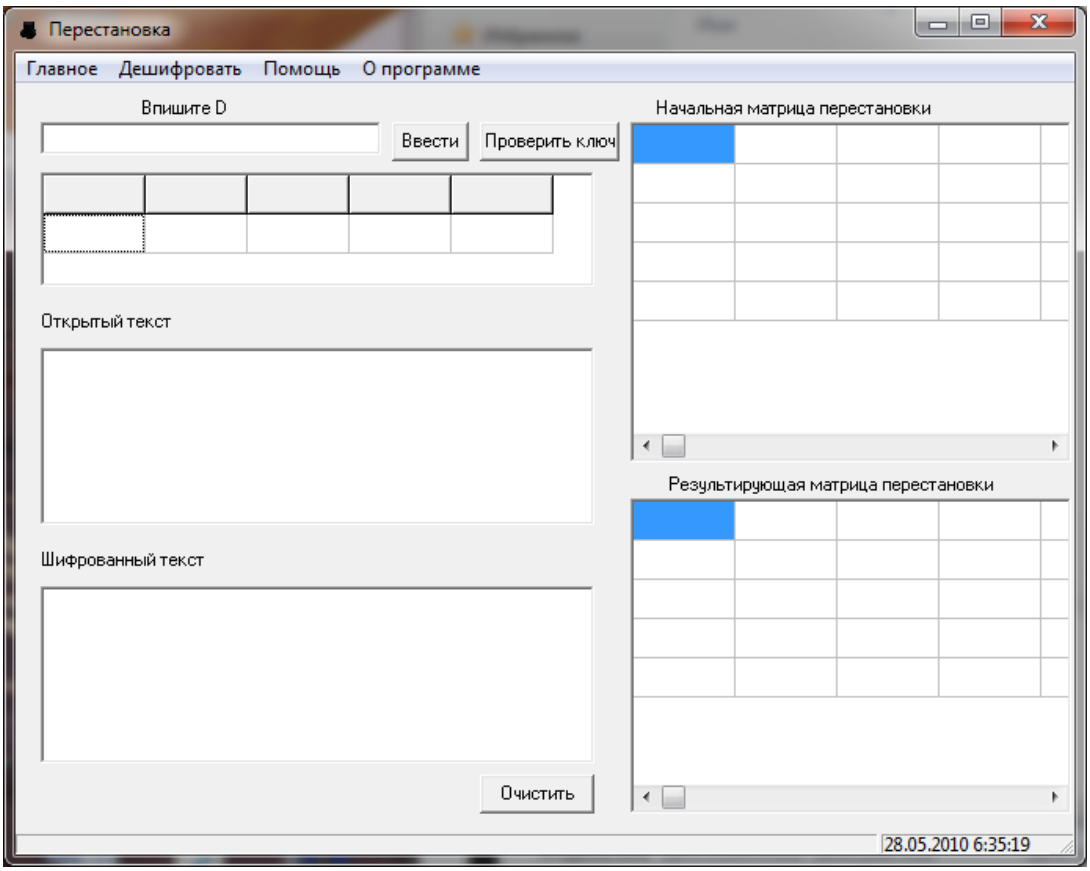
*k (1-2) = k(ОЕ) + k( Я) + k(ОФ) + k(ЦА) + k(ОП) = 6 + 0 + 2 + 5 + 4 =17*
*k (1-3) = k(ОР) + k( И) + k(ОР) + k(ЦИ) + k(ОР) = 8 + 8 + 8 + 7 + 8 = 39*
*k (1-4) = k(ОТ) + k( И) + k(ОН) + k(ЦМ) + k(ОИ) = 8 + 8 +7 + 0 + 6 = 29*

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.

## 2. Описание 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. 10.1.



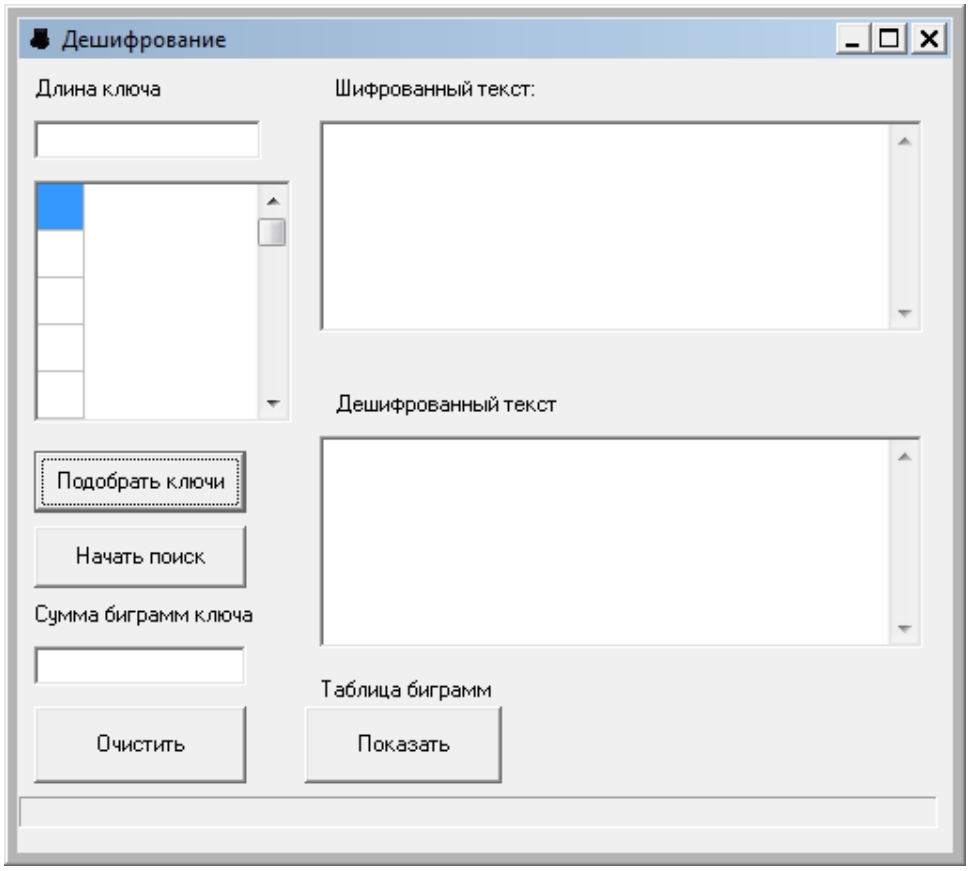




Рис. 10.1. Главное окно программы и окно дешифрования

Дополнительные сведения о программе и авторах программной реализации, представлены на рис. 10.2.

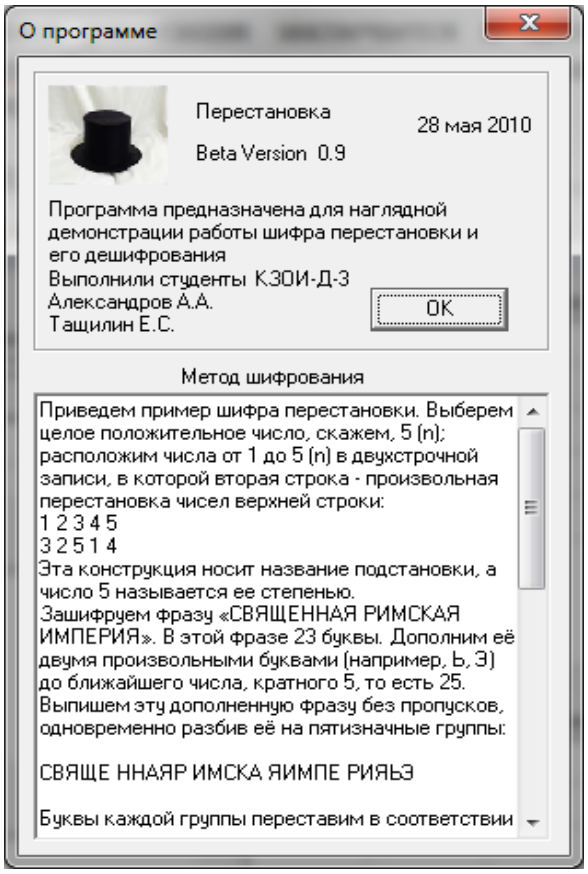



Рис. 10.2. Дополнительные сведения о программе и авторах программной реализации

## 3. Порядок выполнения лабораторной работы

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.

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 вопросы, выбранные в соответствии с номером варианта из *приложения 1*.

*Приложение 1*

| **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?<br>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 2*

## Таблица коэффициентов встречаемости биграмм в тексте

| | АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШТЬЫЪЭЮЯ | Процент |
|---|---|---|
| А | 27867777477788376782667755000679 | 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

# 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***

## 1.Современные методы защиты программного обеспечения

*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 её 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 не дорогой в реализации.

### 1.1. Юридическая защита

Юридическая защита включает в себя такие методы как патентование, оформление авторских прав на интеллектуальную собственность и т.д. Также он предусматривает возможность лицензирования программного продукта, так, например большинство программных продуктов поставляются вместе с лицензией, которая подтверждает право пользователя использовать этот программный продукт, то есть, покупая лицензионную копию программы, пользователь в некой мере производит покупку лицензии на право работы с ее копией.

#### *1.1.1 Международные источники прав авторов программ*

В области авторского права, из ныне действующих на территории России, существует два наиболее общих соглашения. Это Всемирная (Женевская) конвенция об авторском праве от 6 сентября 1952 г. (пересмотрена в Париже 24 июля 1971 г.) и Бернская Конвенция об охране литературных и художественных произведений (Парижский Акт, ВОИС, 24 июля 1971 г.). Непосредственно не упоминая такого объекта права как компьютерная программа, они регулируют отношения в области

авторского права в целом, в частности в области литературных произведений, к которым наше законодательство приравнивает программы

Бернская конвенция вводит ряд основных принципов охраны произведений:

- национальный принцип – предоставление охраны вне зависимости от места первой публикации, основываясь на гражданстве или постоянном проживании автора в стране участнице;
- территориальный принцип – предоставление охраны вне зависимости от гражданства или постоянного пребывания в странах-участниках, основываясь на месте первой (или одновременной) публикации в одной из стран Бернского союза;
- принцип автоматической охраны, согласно которому предоставляемая охрана не должна обуславливаться соблюдением каких-либо формальностей;
- минимальный обязательный (50-летний) срок охраны авторского права исчисляемый со дня смерти автора.

Всемирная (Женевская) конвенция предъявляет менее жесткие требования к охране произведений. Так п. 2 ст. 3 Всемирной конвенции разрешает странам участницам отходить от принципа автоматической охраны и требовать выполнения формальностей для приобретения и реализации авторских прав по отношению ко всем произведениям, впервые опубликованным на их территории, и по отношению к произведениям отечественных авторов, независимо от места их опубликования.

Нормы, касающиеся прав авторов, содержатся также в многочисленных двусторонних и многосторонних соглашениях Российской Федерации, однако в целом они не вносят никаких новых принципов в правовое регулирование данной области.

### *1.1.2. Отечественные источники прав авторов программ*

В Российской Федерации программы для ЭВМ стали относиться к числу объектов интеллектуальной собственности с момента принятия в 1991 г. Основ гражданского законодательства Союза ССР и республик. В большем объеме их юридическая защита была введена Законом РФ от 23 сентября 1992 г. "О правовой охране программ для электронных вычислительных машин и баз данных", вступивший в силу с 20 октября 1992 г. 9 июля 1993 г. был принят Закон РФ "Об авторском праве и смежных правах". Этот Закон, применительно к программам ЭВМ, повторил основные положения Закона о правовой охране программ для ЭВМ, но в некоторых случаях внес определенные уточнения (программы для ЭВМ являются объектами авторского права наряду с литературными произведениями, вопросы истечения срока действия авторского права, вопросы перехода авторского права по наследству и др.). Названные законы соотносятся между собой как общий и специальный.

Предоставляемая данными Законами правовая охрана распространяется на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы, а также базы данных), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код.

Согласно ст.49 Закона "Об авторском праве и смежных правах" и ст.18 Закона о "О правовой охране программ для электронных вычислительных машин и баз данных" правообладатель исключительных авторских и смежных прав вправе требовать от нарушителя:

- признания прав;
- восстановления положения, существовавшего до нарушения права, и прекращения действий, нарушающих право или создающих угрозу его нарушению;

- возмещения убытков, включая упущенную выгоду;
- взыскания дохода, полученного нарушителем вследствие нарушения авторских и смежных прав, вместо возмещения убытков;
- выплаты компенсации в сумме от 10 до 50 000 минимальных размеров оплаты труда (5000-50000 МРОТ в соответствии со ст.18), устанавливаемых законодательством Российской Федерации, в случаях нарушения с целью извлечения прибыли вместо возмещения убытков или взыскания дохода;
- принятия иных предусмотренных законодательными актами мер, связанных с защитой их прав.

В соответствии со ст.146 Уголовного Кодекса РФ предусматривается следующая уголовная ответственность за нарушение авторских и смежных прав:

1. Незаконное использование объектов авторского права или смежных прав, а равно присвоение авторства, если эти деяния причинили крупный ущерб наказываются штрафом в размере от 200 до 400 минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от двух до четырех месяцев, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо лишением свободы на срок до двух лет.

2. Те же деяния, совершенные неоднократно либо группой лиц по предварительному сговору или организованной группой наказываются штрафом в размере от 400 до 800 минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от четырех до восьми месяцев, либо арестом на срок от четырех до шести месяцев, либо лишением свободы на срок до пяти лет.

#### *1.1.3. Патентная защита программ*

Преимущество патентной защиты по сравнению с авторским правом, состоит в одном наиболее принципиальном различии. Так если авторское

право охраняет объективную форму произведения, то патентное право касается содержания тех принципов, которые положены в его основу.

В соответствии с п.1 ст.4 Патентного закона Российской Федерации от 23 сентября 1992 года (с последними изменениями от 7 февраля 2003 г.), изобретению предоставляется правовая охрана, если оно является новым, имеет изобретательский уровень и промышленно применимо.

Российское законодательство применяет критерий мировой новизны. Это означает, что изобретение должно быть неизвестно в мире из общедоступных источников.

Можно выделить несколько важных особенностей патентного права на изобретения:

- права на запатентованную разработку, ограничены национальными рамками соответствующей страны. На один и тот же объект в разных странах патент может быть выдан разным лицам.
- права, вытекающие из патента, действуют лишь определенный промежуток времени. В соответствии с российским законодательством патент на изобретение действует в течение 20.
- на патентообладателя возлагается обязанность использовать свой патент и уплачивать ежегодные пошлины за его поддержание.
- стоимость составления, выдачи и поддержания патента, может быть очень существенной как для индивидуального патентообладателя, так и для малой российской фирмы. Расходы на патент, напрямую зависят от количества стран, в которых он будет охраняться.

Хотя в п.2 ст.4 Патентного закона прямо сказано, что "не признаются патентоспособными изобретениями программы для вычислительных машин" применение данного института к защите отдельных видов программ и алгоритмов все же возможно.

Так, например, в составе некоторого совместимого с компьютером устройства могут быть запатентованы программы-драйвера, также в качестве способа, возможно патентование некоторых алгоритмов.

### *1.1.4. Зашита программ институтом коммерческой тайны*

В п.2 ст.3 Федерального закона ”О коммерческой тайне” от 29 июля 2004 года № 98-ФЗ (с изменениями на 24 июля 2007 года), коммерческая тайна определяется, как сведения любого характера (производственные, технические, экономические, организационные и другие), в том числе о результатах интеллектуальной деятельности в научно-технической сфере, которые имеют действительную или потенциальную коммерческую ценность в силу неизвестности их третьим лицам, к которым у третьих лиц нет свободного доступа на законном основании и в отношении которых обладателем таких сведений введен режим коммерческой тайны.

Стоимость исходного кода программы, как правило, во много раз выше стоимости прав на объектный код. По этой причине разработчик, зарабатывающий на распространении объектного кода, обычно держит исходный код в тайне. Поскольку исходный код содержит информацию о структуре программы, принципах ее работы и заложенных в нее алгоритмах, то он представляет коммерческую ценность в силу неизвестности ее третьим лицам.

В соответствии с п.1 ст.183 Уголовного кодекса РФ за собирание сведений, составляющих коммерческую тайну, любым незаконным способом предусматривается уголовная ответственность – штраф в размере до восьмидесяти тысяч рублей или в размере заработной платы или иного дохода осужденного за период от одного до шести месяцев либо лишение свободы на срок до двух лет. В данном случае наказуемы сами действия, независимо от того, повлекли ли они какие-либо последствия.

В соответствии с п.2 ст.183 УК незаконное разглашение или использование сведений, составляющих коммерческую тайну, без согласия их владельца лицом, которому она была доверена или стала известна по службе или работе наказывается штрафом в размере до ста двадцати тысяч рублей или в размере заработной платы или иного дохода осужденного за период до одного года с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет либо лишением свободы на срок до трех лет.

Те же деяния, причинившие крупный ущерб или совершенные из корыстной заинтересованности, наказываются штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет либо лишением свободы на срок до пяти лет.

### *1.1.5. Лицензирование программ*

В соответствии с п.86 ст.17 Федеральный закон от 8 августа 2001 г. N 128-ФЗ "О лицензировании отдельных видов деятельности" обязательному лицензированию подлежит “деятельность по изготовлению экземпляров аудиовизуальных произведений, программ для электронных вычислительных машин (программ для ЭВМ), баз данных и фонограмм на любых видах носителей”.

*Лицензирование программ* – это процедура, позволяющая организации или частному лицу приобрести, установить и использовать программное обеспечение на отдельном компьютере или в сети, соответственно лицензионному соглашению с производителем этого программного обеспечения. Цель лицензирования – защитить как инвестиции компании-разработчика, минимизировав вероятность его взлома пиратами, так и

инвестиции предприятия, снизив риск наказания за использование пиратского программного обеспечения. Как правило, разработчик реализует лицензионное соглашение путем встраивания в продукт специальных механизмов, не позволяющих использовать программу в случае нарушения пользователем каких-либо пунктов этого соглашения. К примеру, запуск демонстрационной версии программы по истечении срока ее использования обычно приводит к тому, что программа (или, по крайней мере, какие-то из ее возможностей) становится недоступной.

Примером лицензирования программ является лицензия GNU General Public License (Универсальная общедоступная лицензия GNU или Открытое лицензионное соглашение GNU) – самая популярная лицензия на свободное программное обеспечение, созданная в рамках проекта GNU. Цель GNU GPL – предоставить пользователю права копировать, модифицировать и распространять (в том числе на коммерческой основе) программы (что по умолчанию запрещено законом об авторских правах), а также гарантировать, что и пользователи всех производных программ получат вышеперечисленные права.

GPL предоставляет получателям компьютерных программ следующие права, или "свободы":

- свободу запуска программы, с любой целью;
- свободу изучения того, как программа работает, и ее модификации (предварительным условием для этого является доступ к исходному коду);
- свободу распространения копий;
- свободу улучшения программы, и выпуска улучшений в публичный доступ (предварительным условием для этого является доступ к исходному коду).

## 1.2. Техническая защита

Существует множество технических методов защиты программных продуктов. Ниже будут рассмотрены, наиболее интересные и актуальные из них.

### *1.2.1 Использование парольной защиты (кода активации)*

Этот вариант, на сегодня, является самым распространённым. Принцип действия защиты основан на идентификации и аутентификации пользователя ПО путём 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 не запуске ПО, в зависимости от результатов проверки.

### *1.2.2 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.

### *1.2.3 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.11.1).

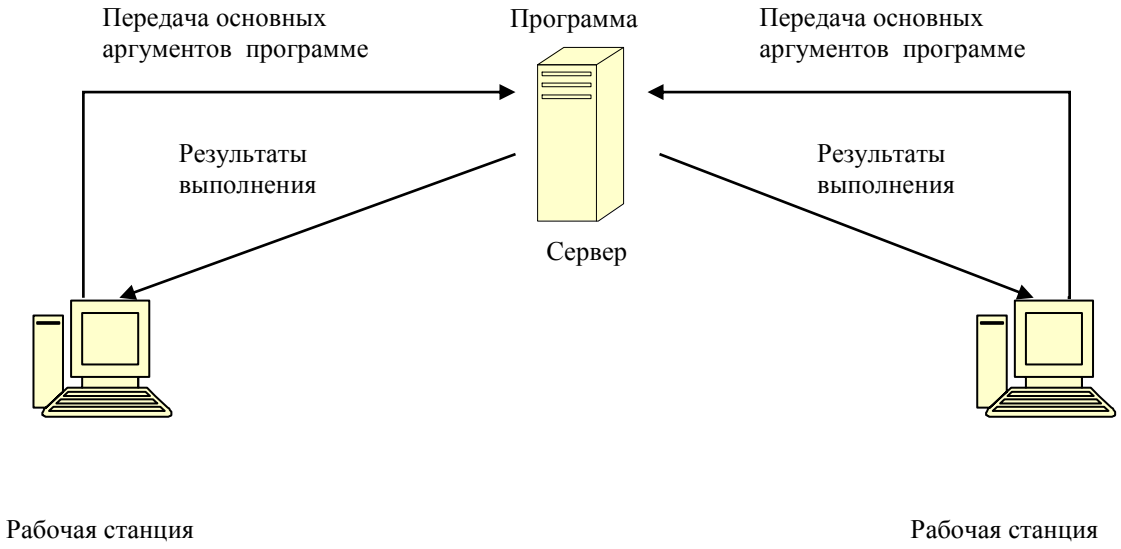



Рис. 11.1. Выполнение программы на стороне сервера

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.

### *1.2.4 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.11.2).

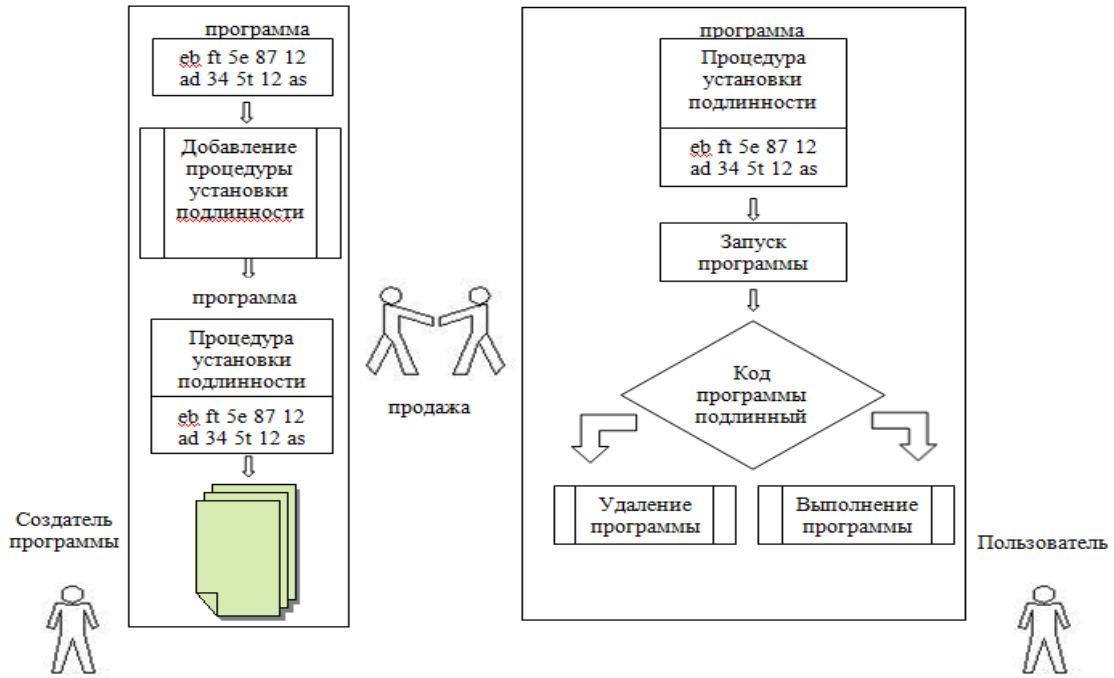



Рис. 11.2. Установка подлинности кода

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, то есть 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.

### *1.2.5. 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 зашифровывание кода программы, после чего она в зашифрованном виде поставляется конечным пользователям (иногда эффективно зашифровывать только наиболее важные, критические, участки кода, а не весь код программы). Когда пользователь запускает такую программу, вначале будет запущена процедура расшифровки программы, которой потребуется ключ, с помощью которого и будет расшифрована запускаемая программа (рис.11.3).

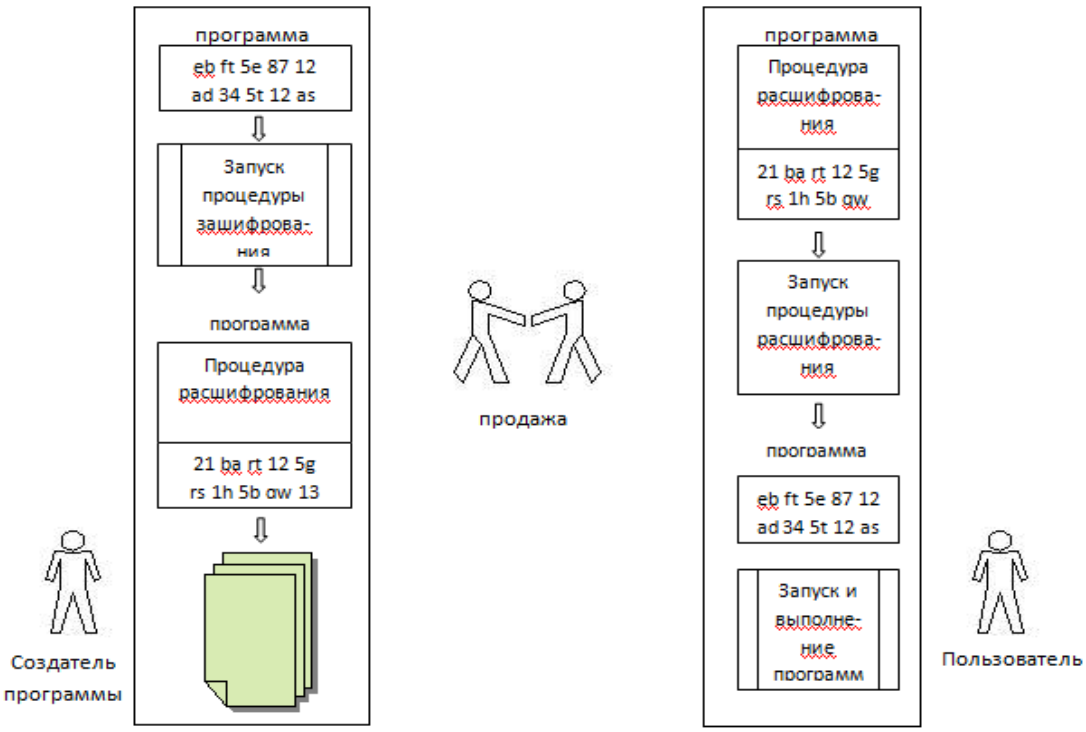



Рис. 11.3. Шифрование программного кода

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.

### *1.2.6. 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, то есть 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 не достаточно, 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.11.4).

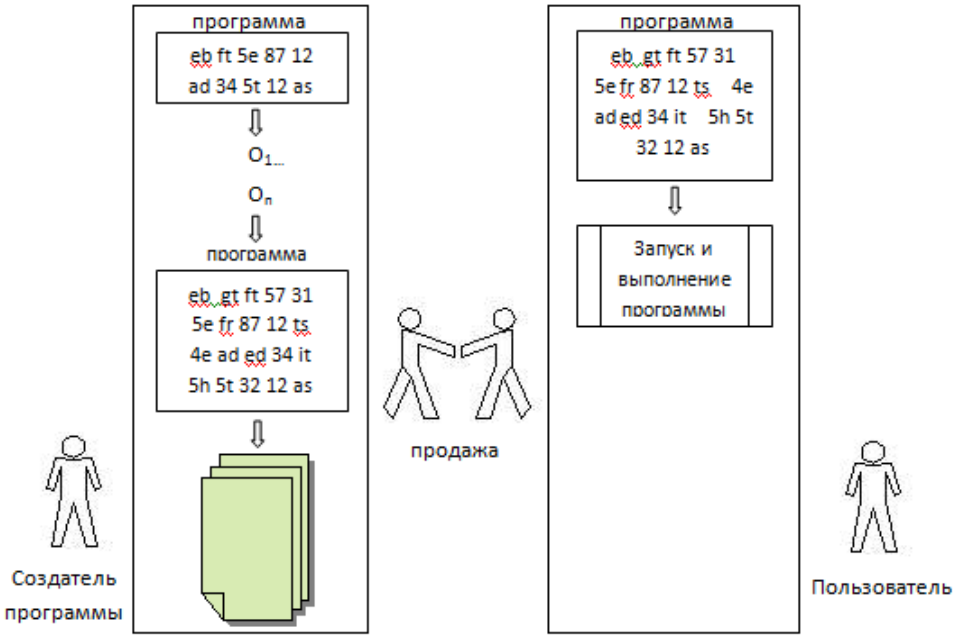



Рис. 11.4. Обфускация программного кода

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 (то есть 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.

### *1.2.7. 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.

### *1.2.8. 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 определённому интерфейсу компьютера, далее защищённая 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 определённым 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 путём 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.

# 2.Основные форматы исполняемых файлов. Специфика PE-файлов

## 2.1. История формата исполняемых файлов

За время существования персональных компьютеров на процессорах семейства x86 (начиная от IBM PC XT на процессоре Intel 8086) успело смениться несколько форматов двоичных файлов, предназначенных для хранения откомпилированного кода программы.

В операционной системе DOS (Disk Operating System) поддерживалось два основных формата исполняемых файлов: COM (от английского Command) и EXE (от английского Executable). COM-файлы загружались в оперативную память без каких-либо дополнительных настроек, и их размер не должен был превышать 64 Кбайта. EXE-файлы не имели таких жестких ограничений на размер и состояли из заголовка, включающего всю необходимую информацию для правильной загрузки программы в память, и собственно кода программы. В начале заголовка DOS-овских EXE-файлов всегда находятся два байта 0x4D 0x5A (символы ASCII M и Z), по этой причине этот заголовок часто называют MZ-заголовком (MZ Header). Буквы MZ являются инициалами Марка Збыковски (Mark Zbikowski), являвшегося разработчиком данного формата. Сейчас все исполняемые файлы содержат MZ-заголовок, за которым может следовать информация о другом формате. С появлением 16-битовой версии Windows возникла потребность в расширенном формате исполняемых файлов. В Windows была реализована поддержка динамически подсоединяемых библиотек (dynamic-link library. DLL), поэтому новый формат должен был обеспечить возможность хранения, в частности, таблиц экспортируемых (находящихся в DLL и доступных другим модулям) и импортируемых (находящихся во внешних библиотеках) функций. Кроме этого в Windows широко используются ресурсы – двоичные данные, содержащие иконки, курсоры, описания

диалогов и т. д., которые желательно хранить внутри исполняемых файлов. Все актуальные на тот момент требования были учтены при разработке формата, получившего название New Executable (NE, новый исполняемый файл). Заголовок такого файла начинается с символов 'NE'.

Для хранения драйверов виртуальных устройств Windows (Virtual device driver, VxD) применялся формат Linear Executable (LE, линейный исполняемый файл). Его модификация под названием Linear executable (LX) использовалась для хранения исполняемых файлов, используемых в операционной системе OS/2 начиная с версии 2.0.

В связи с появлением 32-битовой операционной системы Windows NT в Microsoft был разработан формат Portable Executable (PE, переносимый исполняемый файл). Точнее, был доработан до своих нужд взятый за прототип формат объектных файлов COFF (Common Object File Format), используемый в Unix. Слово "переносимый" в названии, скорее всего, отражает тот факт, что один и тот же формат файла использовался как во всех 32-битовых операционных системах Microsoft на платформе x86, так и в Windows NT на других платформах (MIPS, Alpha и Power PC). Этот формат является основным для всех современных версий Windows и именно он будет рассматриваться в дальнейшем.

### *2.2. Структура PE-файлов*

Типичный исполняемый PE-файл имеет следующую структуру (рис. 11.5).

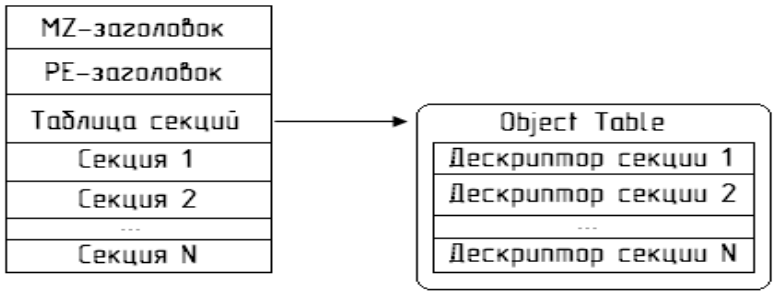



Рис. 11.5. Обобщённая структура PE-файла

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 определенна 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. Т.е. если файл занимает 1 байт, с диска будет все равно прочитано 512 (1 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 секциях не бывает пустых мест. Каждая стандартная секция имеет определенное назначение, например:

- в ".text" распложен 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.

# 3.Техника внедрения кода в PE-файлы

Существуют следующие способы внедрения:

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.

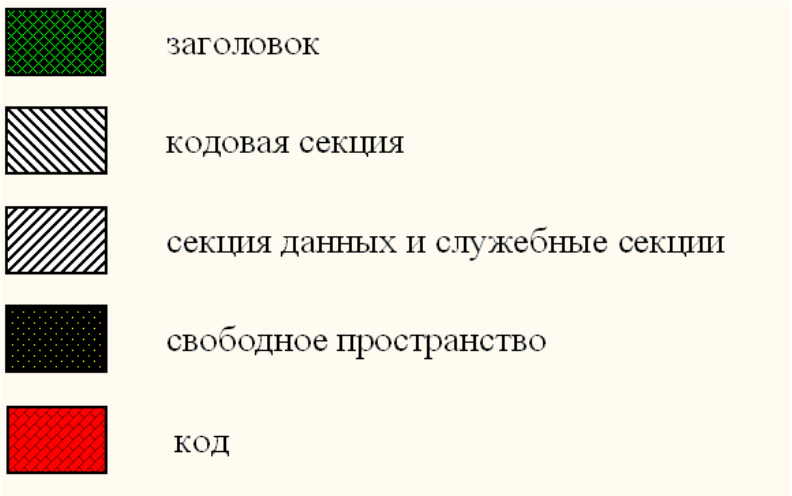



Рис. 11.6. Условные графические обозначения, принятые в дальнейшем изложении

## 3.1. Классификация механизмов внедрения

Механизмы внедрения можно классифицировать по-разному: по месту (начало, конец, середина), по "геополитике" (затирание исходных данных, внедрение в свободное пространство, переселение исходных данных на новое место обитания), по надежности (предельно корректное,

вполне корректное и крайне некорректное внедрение), по 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.

### *3.1.1 Категория A: внедрение в пустое место файла*

Проще всего внедриться в пустое место файла. На сегодня таких мест известно три: а) PE-заголовок; б) хвостовые части секций; в) 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 ей часть, если размер не позволяет большего (рис.11.7.).


Рис. 11..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. 11.8): 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 .

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 внедрённый код.

На рис.11.8 показан типичный PE-заголовок файла, а на рис. 11.9 тот же файл после внедрения информации.

Рис. 11.8. Типичный PE-заголовок файла

Рис. 11.9. 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.11.10).

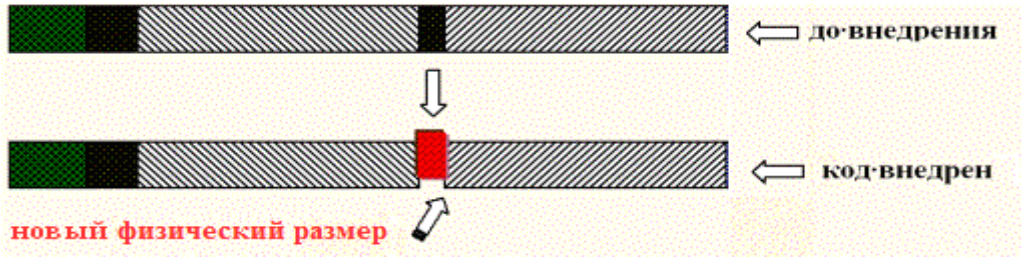



Рис. 11.10. Внедрение кода в хвост секции, оставшийся от выравнивания

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.11.11.

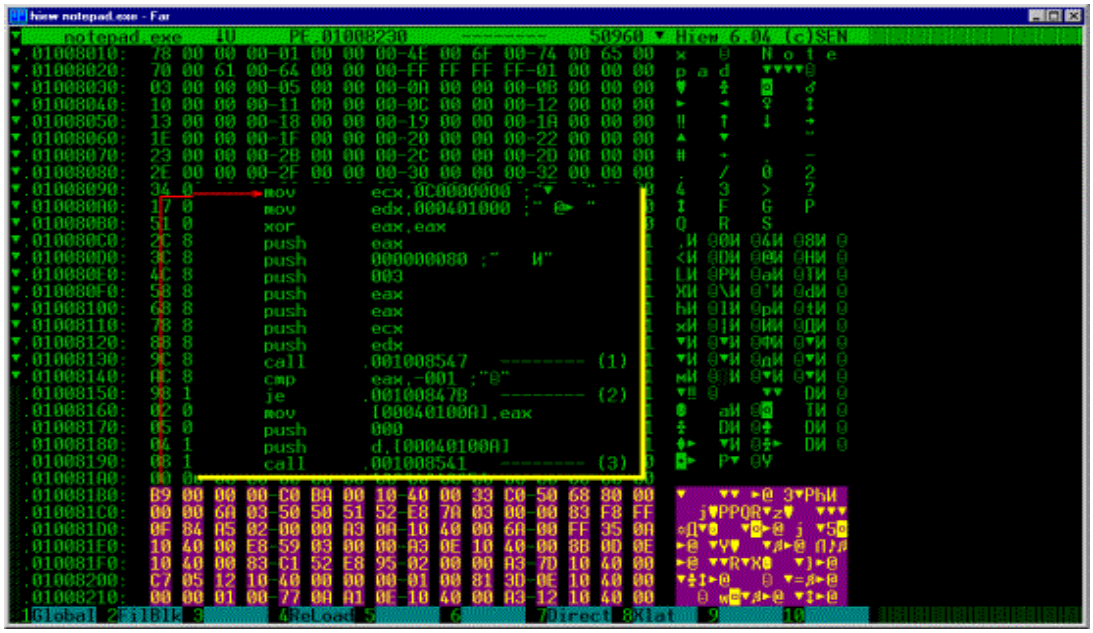


Рис 11.11. Осмысленный машинный код в хвосте секции данных

*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. 11.12.

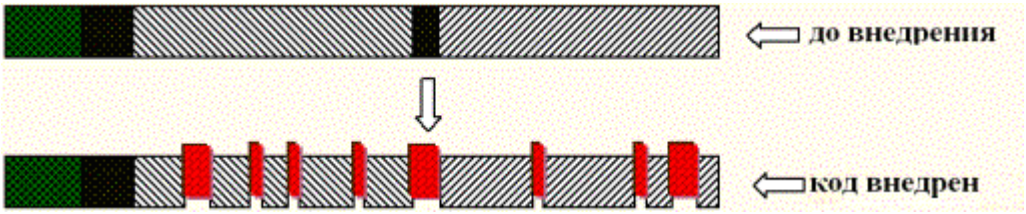



Рис. 11.12. Внедрение кода в регулярные цепочки

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, т.к. 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.

### *3.1.2. 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.11.13).

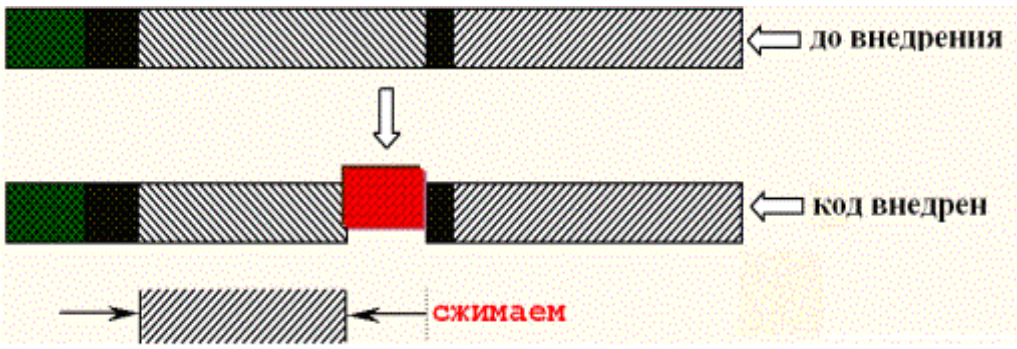



Рис. 11.13. Внедрение кода путем сжатия секции

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.

### *3.1.3. 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 заголовка (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, т.к. 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.11.14).

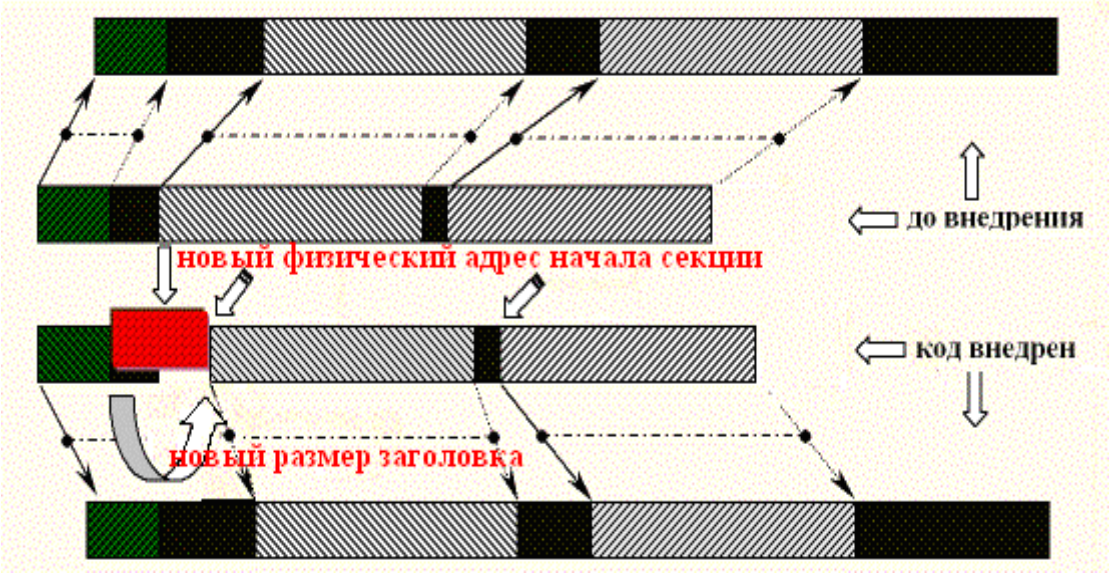



Рис. 11.14. Исполняемый файл и его проекция в память до и после внедрения кода путем раздвижки заголовка

Данный метод внедрения распознается аналогично обычному методу внедрения в PE-заголовок.

# 4. Описание программной реализации встраивания ЦВЗ

## 4.1 Описание реализации

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.11.15.


Рис. 11.15 Алгоритм встраивания ЦВЗ

### 4.2 Перечень основных программных модулей

- Unit1.cpp – представляет собой главный модуль программы, который содержит основную рабочую форму, функции загрузки файлов, встраивания и извлечения ЦВЗ;

- Unit2.cpp – содержит базовую форму ввода пароля и функцию получения хэш-значения пароля;

- Unit3.cpp – справка о программе;

- Filigrana_Справка.chm – содержит скомпилированную справку по использованию программы;

- Unit1.h, Unit2.h и Unit3.h – содержат шаблонны функций и список использованных переменных в Unit1.cpp и Unit2.cpp, Unit3.cpp соответственно;

- Project1.cpp является файлом-проекта данной программы.

### 4.3 Выходные данные

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 MB (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 чём нам и сообщает программа (рис. 11.16).

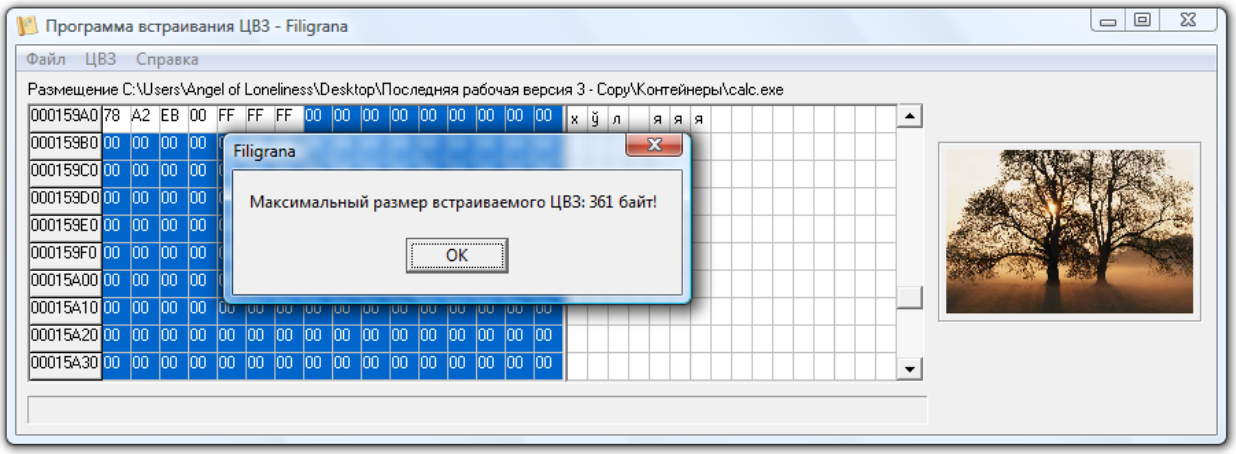



Рис. 11.16. Сообщение о допустимом размере встраиваемого ЦВЗ

Для второго файла возможный размер ЦВЗ для встраивания увеличился до 13668 байт (рис. 11.17).

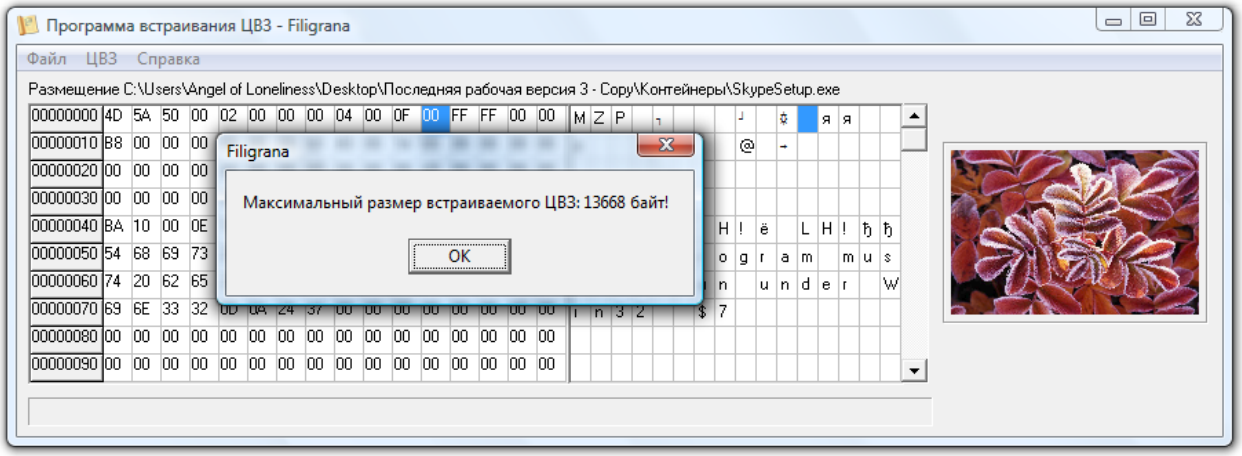



Рис. 11.17. Сообщение о допустимом размере встраиваемого ЦВЗ

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.11.18).

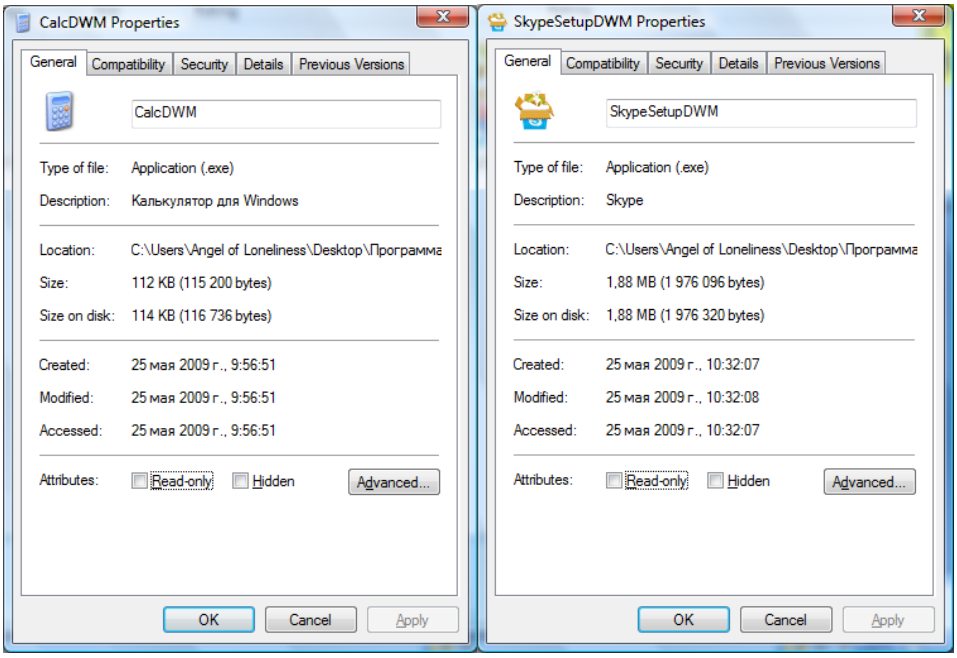
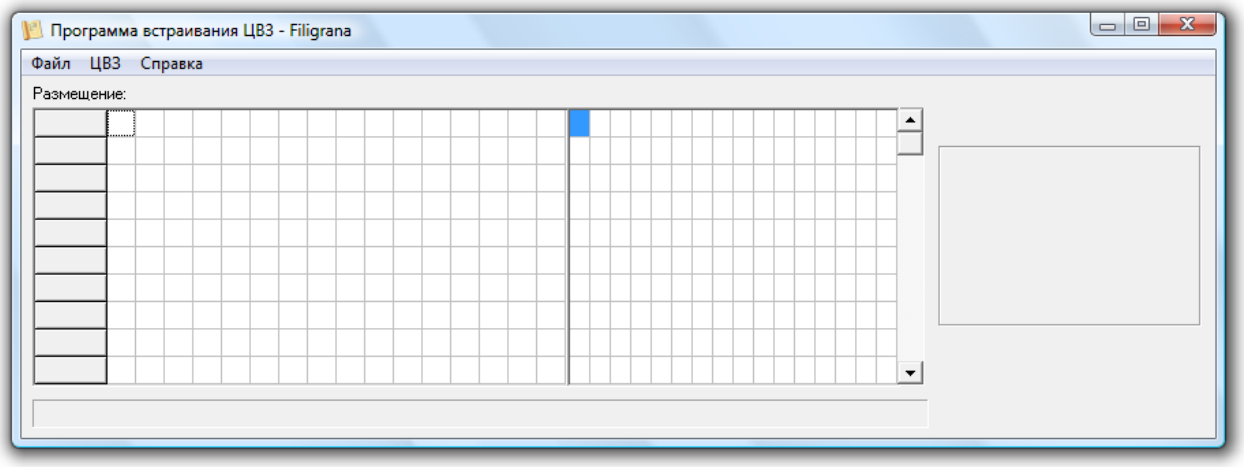



Рис. 11.18. Размеры exe-файлов, содержащих ЦВЗ

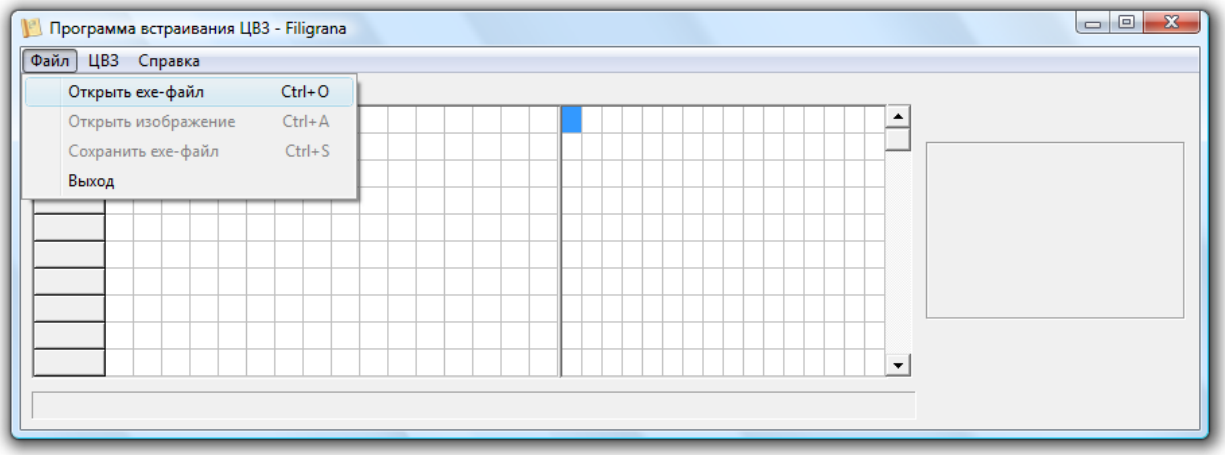
## 5. Порядок выполнения лабораторной работы

### Встраивание цифрового водяного знака

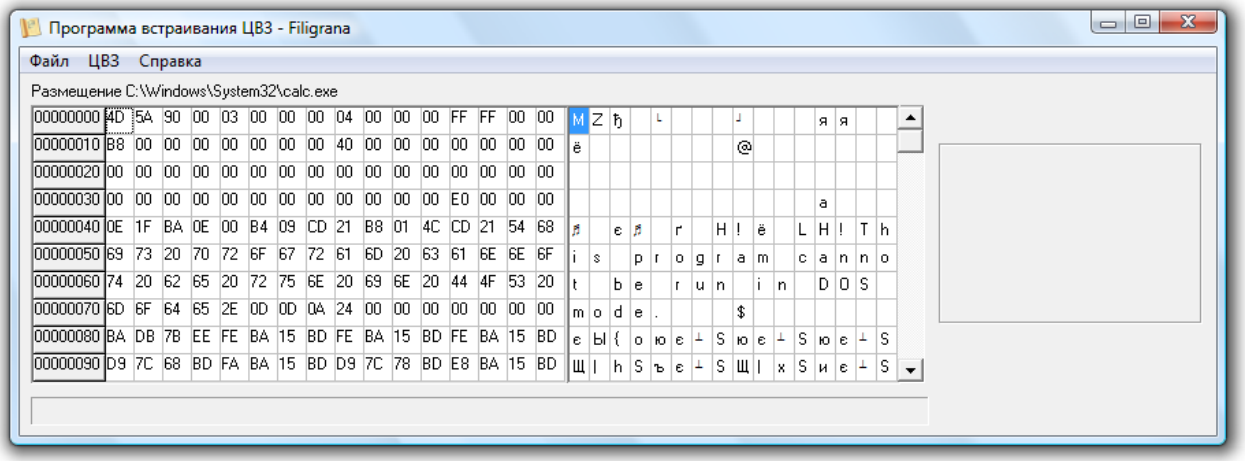
1. Запустите программу Feligrana, предварительно установленную на ваш компьютер.



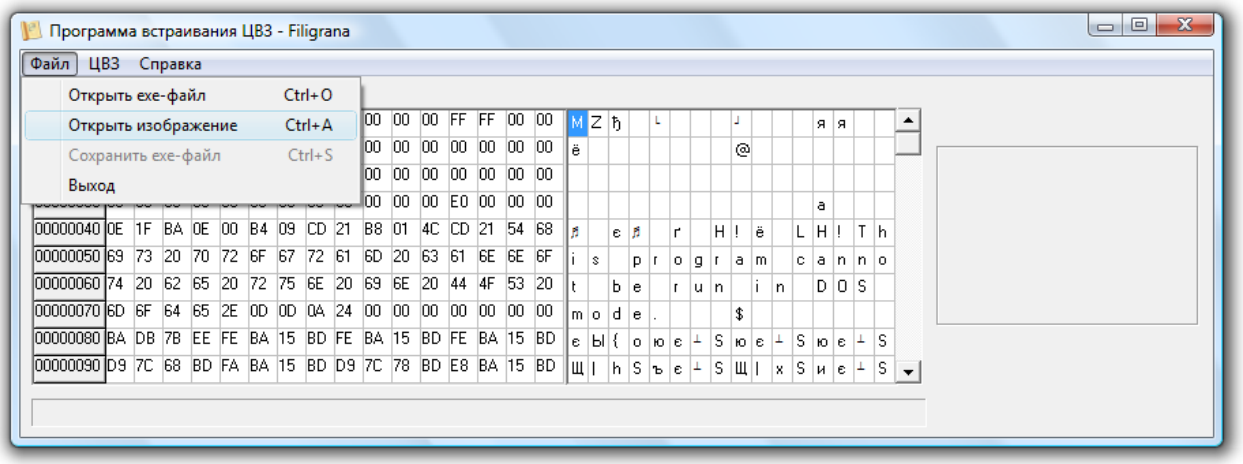
2. Выберете пункт "Открыть exe-файл" из меню "Файл" появившегося окна или нажмите комбинацию клавиш "Ctrl+O".



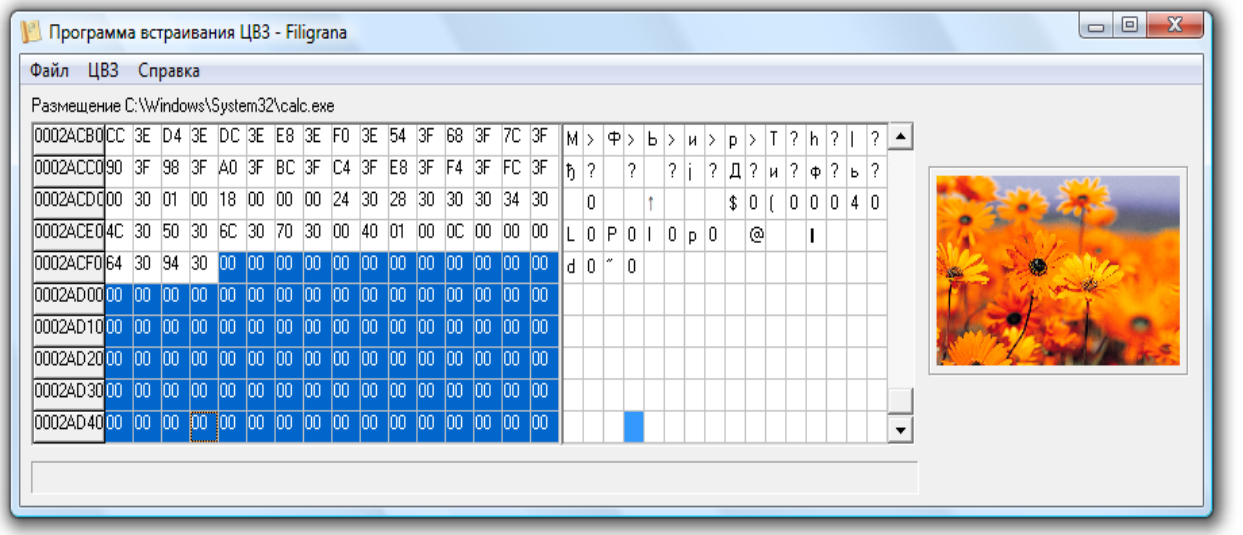

3. В появившемся диалоговом окне выберете необходимый exe-файл, два раза кликнув на нём левой кнопкой мыши или с помощью кнопки "Открыть". В окне главной формы отобразятся данные о выбранном файле в шестнадцатеричном и ASCII формате.



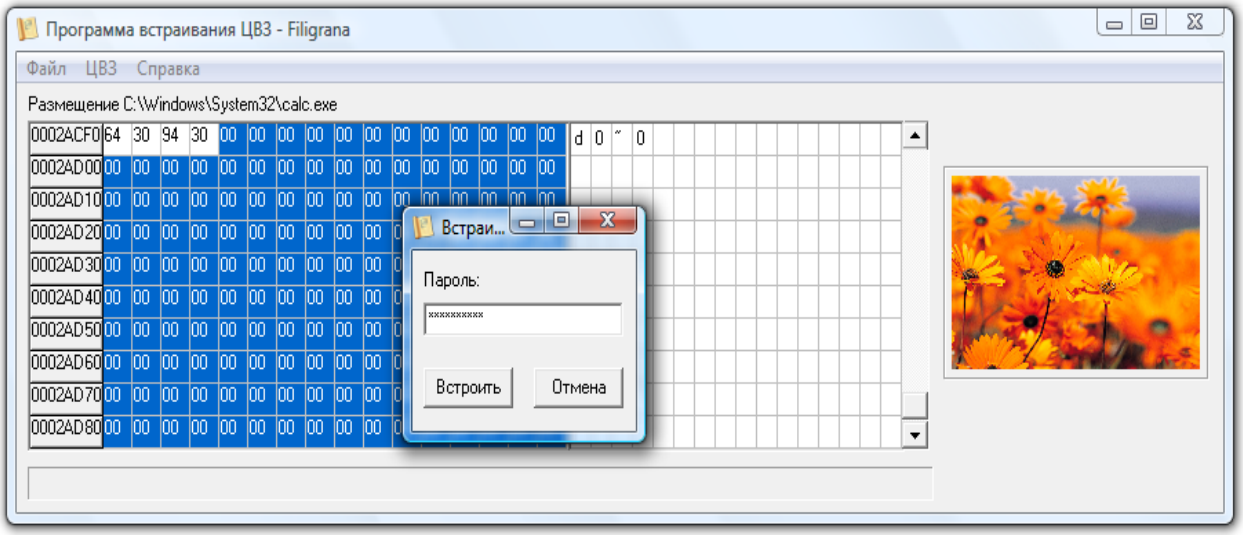

4. Выберете пункт "Открыть изображение" из меню "Файл" или нажмите комбинацию клавиш "Ctrl+A".



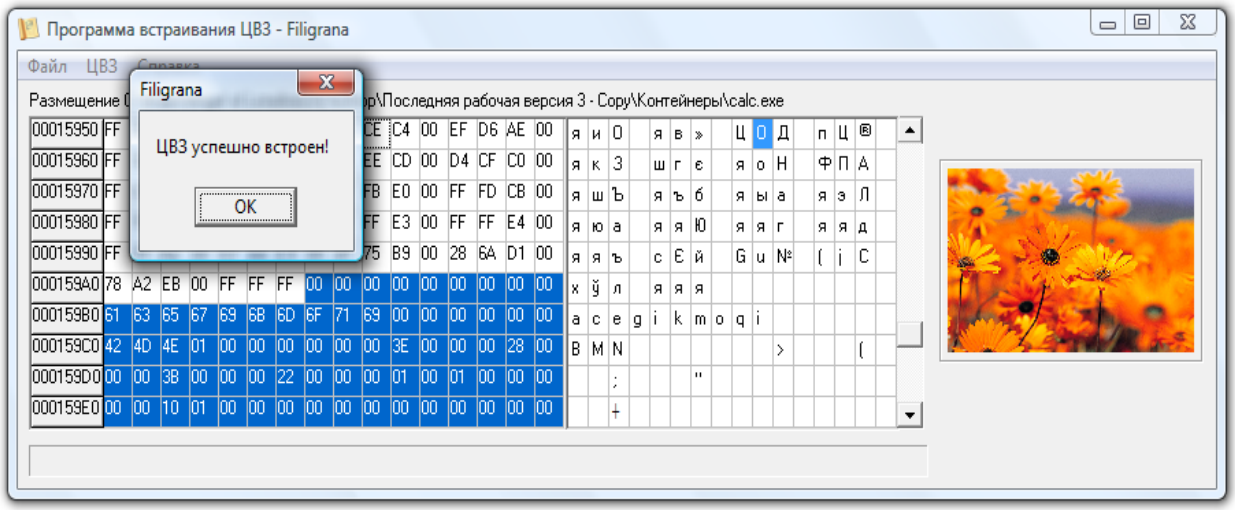

5. В появившемся диалоговом окне выберете изображение в формате bmp, которое вы хотите использовать в качестве ЦВЗ, два раза кликнув на нём левой кнопкой мыши или с помощь кнопки "Открыть". В окне preview главной формы отобразится выбранное изображение, в hex-окне курсор автоматически переместится на строку, с которой начнётся последующее встраивание цифрового водяного знака.




6. Для того, 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.

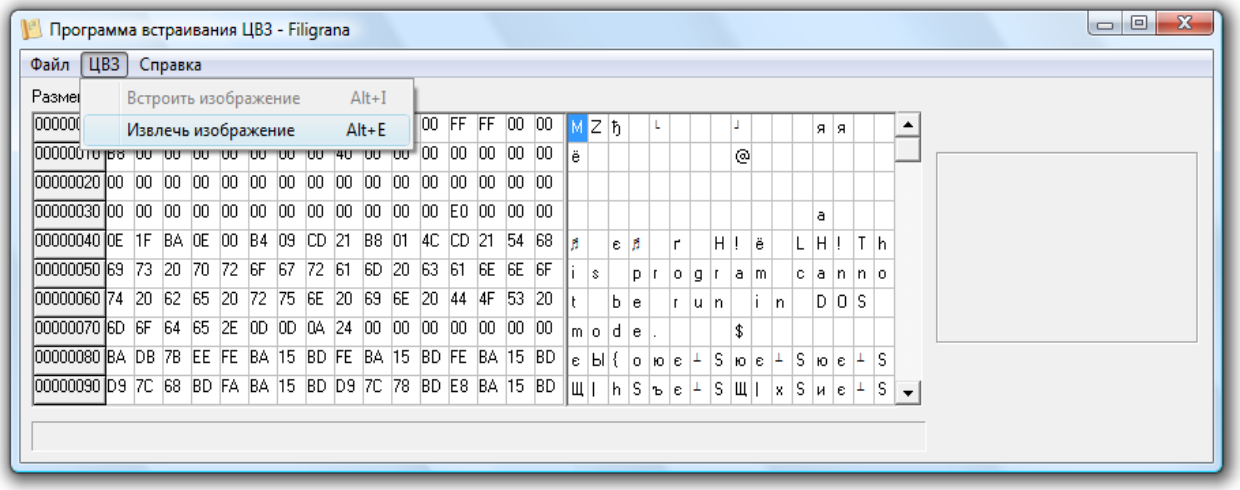



8. Для сохранения исполняемого файла со встроенным ЦВЗ выберете пункт "Сохранить exe-файл" из меню "Файл" или нажмите комбинацию "Ctrl+S", введите имя сохраняемого файла, и нажмите кнопку "Сохранить".

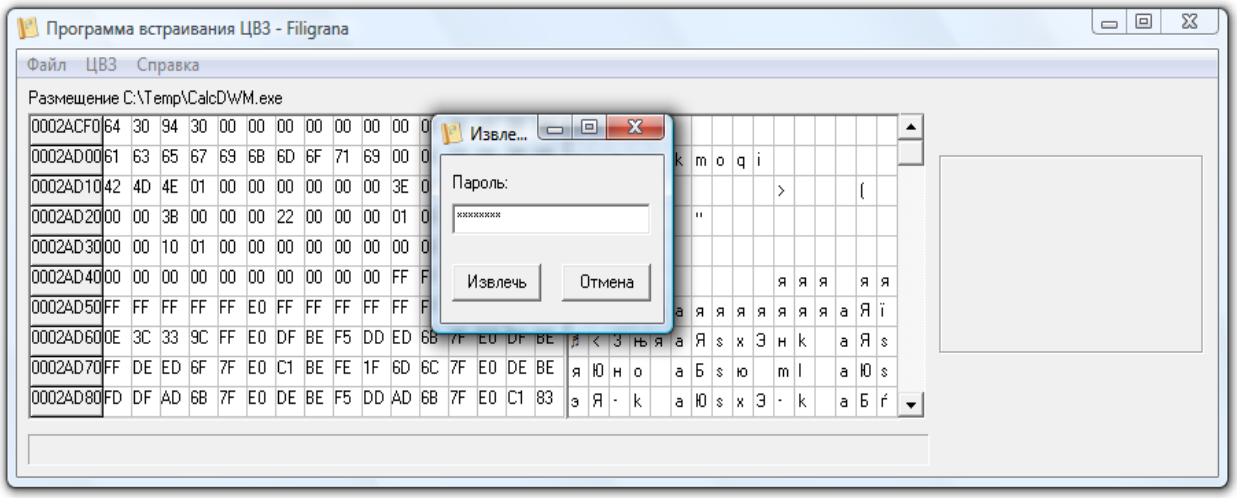
***Внимание!*** *Для успешного извлечения ЦВЗ необходимо запомнить свой пароль или сохранить его в месте, недоступном для других пользователей. Никому не показывайте свой пароль!*

### Извлечение цифрового водяного знака

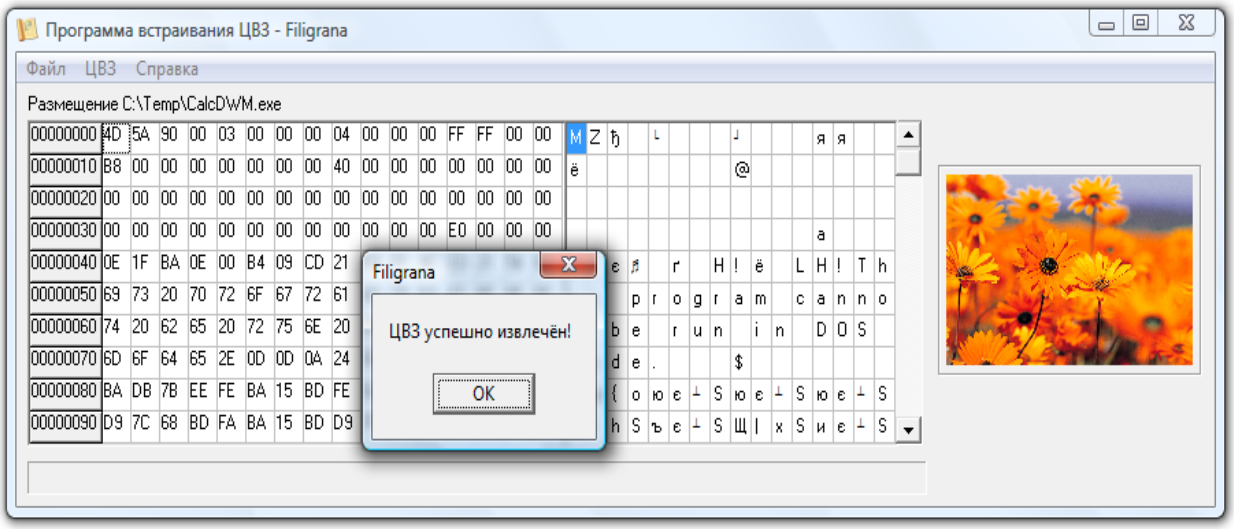
1. Откройте exe-файл, содержащий ЦВЗ. Выберете пункт "Извлечь изображение" меню "ЦВЗ" главной формы или нажмите комбинацию клавиш "Alt+E".




2. Для извлечения изображения в окне "Извлечение ЦВЗ" введите пароль, который использовался при встраивание 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.




### Требования к отчету

Сохранить в отчете экранные формы, демонстрирующие процесс встраивания ЦВЗ в различные исполняемые файлы. Сделать выводы об эффективности изученного метода защиты.

Включить в отчет о лабораторной работе **ответы на контрольные вопросы**, выбранные в соответствии с номером варианта из *приложения 1*.

*Приложение 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 №12

# 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_2.02*** *(исполняемые файлы находятся в папке* **Программные модули***)*

## 1. Цифровые водяные знаки

*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.12.1) – 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. 12.1. 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 не только отразить 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, то есть 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.

Несмотря на многочисленные достоинства ЦВЗ во многих странах Америки и Европы, в Российской Федерации Федерального закона (ФЗ) «О цифровом водяном знаке» еще не принималось и даже не существует его проекта. Конечно, можно подвести ЦВЗ под статью 1299 ГК РФ «Технические средства защиты авторских прав», в которой говорится о том, что «техническими средствами защиты авторских прав признаются любые технологии, технические устройства или их компоненты, контролирующие доступ к произведению, предотвращающие либо ограничивающие осуществление действий, которые не разрешены автором или иным правообладателем в отношении произведения». Однако для Знака охраны авторского права выделена целая статья в Гражданском Кодексе (ст. 1271 «Знак охраны авторского права»), а электронная цифровая подпись (ЭЦП) утверждена многими нормативными актами, и в 2002 г. принят ФЗ «Об электронной цифровой подписи», который обеспечивает «правовые условия использования электронной цифровой подписи в электронных документах, при соблюдении которых электронная цифровая подпись в электронном документе признается равнозначной собственноручной подписи в документе на бумажном носителе». Получается, что из всех существующих в настоящее время способов

защиты авторского права, только ЦВЗ не имеет в России законного статуса.

Основные области использования технологии ЦВЗ могут быть объединены в три группы: *защита от копирования*, *скрытая аннотация документов* и *доказательство аутентичности информации (аутентификация)*, (рис.12.2).

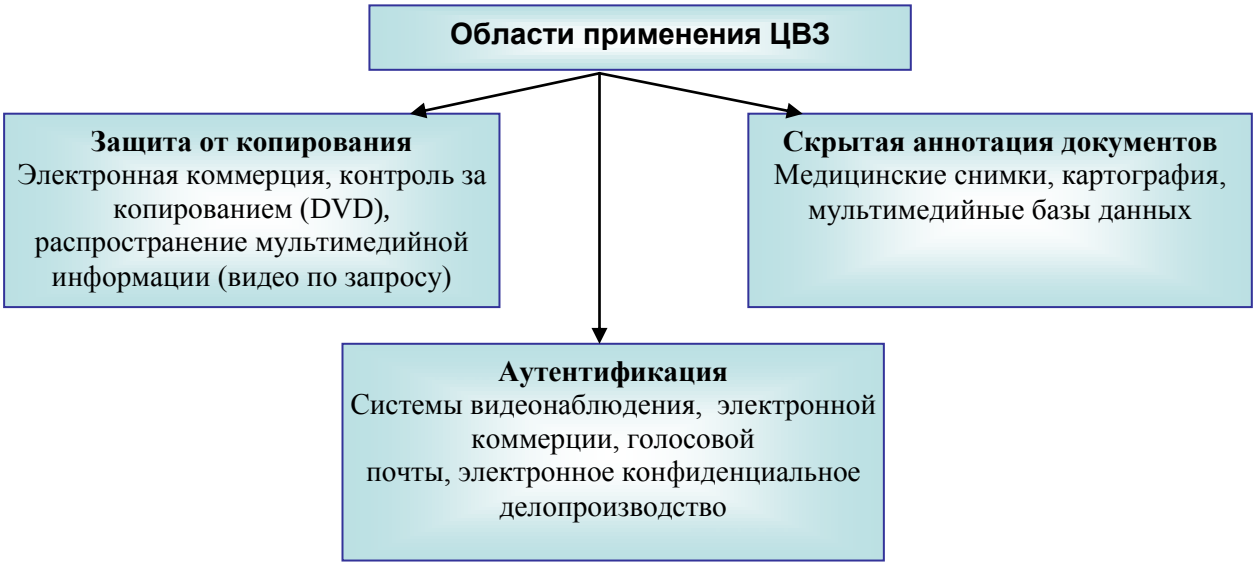



Рис. 12.2. Области применения ЦВЗ

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.

# 2. 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.

## 2.1. 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 коэффициентов ДКП, также размером 8×8 (рис.12.3). 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.

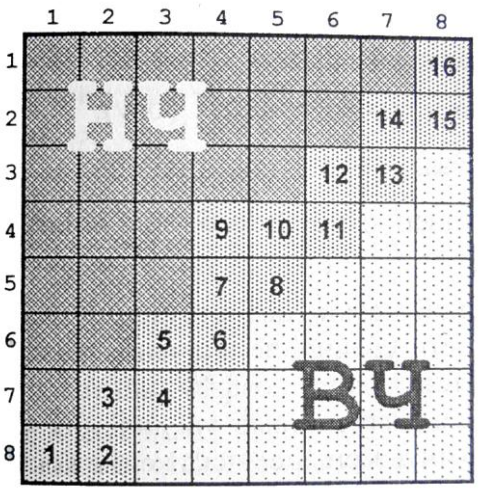



Рис.12.3. Матрица ДКП

**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.12.4). Для большей стойкости и скрытности результатов использования данного стеганографического метода количество встраиваемой информации на практике пытаются уменьшить.


Рис. 12.4. Внедряемый ЦВЗ

Перед встраиванием изображение подвергается случайным перестановкам, или перестановкам, зависящим от характеристик блоков

контейнера, в который будет встроена информация (рис.12.5). ЦВЗ встраивается в среднечастотные коэффициенты ДКП (четвертая часть от общего количества). Эти коэффициенты расположены вдоль второй диагонали матрицы ДКП.

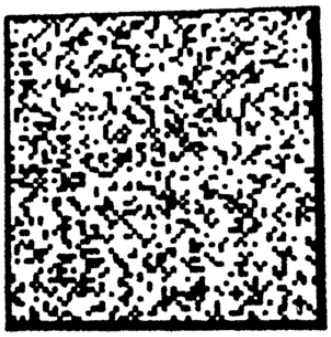


Рис. 12.5. Результат псевдослучайной перестановки элементов ЦВЗ

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:

$$d_1(i) = sign\left(c_b(j,k) - c_{b-1}(j,k)\right).$$

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. 12.6.


Рис. 12.6. 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 НЧ 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:

$$t_0 = 1,\ t_{i+1} = \frac{1+\alpha}{1-\alpha} t_i,$$

где параметр $\alpha \in (0,1)$ строится функция

$$ind(t) = (-1)^i, \qquad t>1,\ \tau_i \le t < \tau_{i+1},$$

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 $\alpha t$. На рис. 12.7 показан вид функции $ind(t)$ для $\alpha = 0.1,\ \alpha = 0.2,\ \alpha = 0.3.$

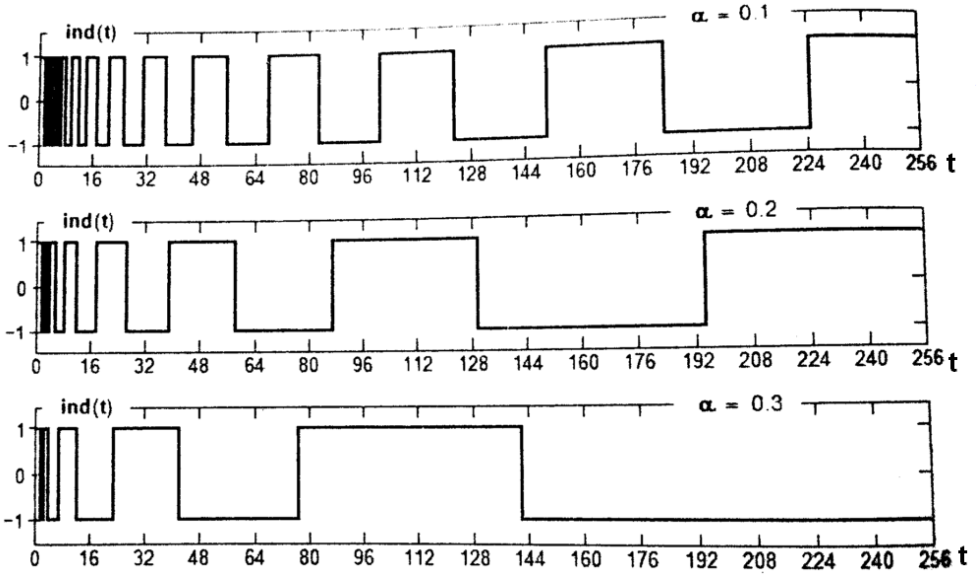



Рис. 12.7. Индексная функция $ind(t)$

Для внедрения бита ЦВЗ $s_i$ в коэффициент $c_j$ последний изменяется не более чем на $100\alpha$ процентов так, чтобы $ind(|c_j'|) = s_i$.

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 $\alpha$ и сложения результата со значением коэффициента. Предварительное кодирование ЦВЗ выполняется по следующему алгоритму.

Вход алгоритма: сообщение длины $M$, состоящее из символов $m_i \in \{1,...,B\}$.

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 $m_i$ 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×N пикселей. Разделим его на множество непересекающихся блоков $B_i$, каждый размера 3×3, где $1 \le i \le (M*N)/(3*3)$. 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.

$$Цвет \quad кандидата = \begin{cases} черный, \quad если \quad N_B < 4 \\ белый, \quad если \quad N_B \geq 7 \\ любой, \quad если \quad 4 \leq N_B < 7 \end{cases},$$

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:

$$r(m,n) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} (\omega(i,j) \times I_{mn}(i,j)),$$

$$I_{mn}(i,j) = \begin{cases} 0, \quad если \quad O(m,n) = O(m+i, n+j) \\ 1, \quad если \quad O(m,n) \neq O(m+i, n+j) \end{cases},$$

где $O(m,n)$ – значение функции яркости пикселя исходного изображения с координатами *(m, n)*;

$$W = (\omega_{i,j}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 0 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \text{ – матрица веса.}$$

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 рис.12.8 представлены увеличенный фрагмент оригинального изображения и фрагмент того же самого изображения после внедрения в него ЦВЗ объемом 40 бит.

Рис. 12.8. 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 малыми 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, то есть 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 К:

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

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

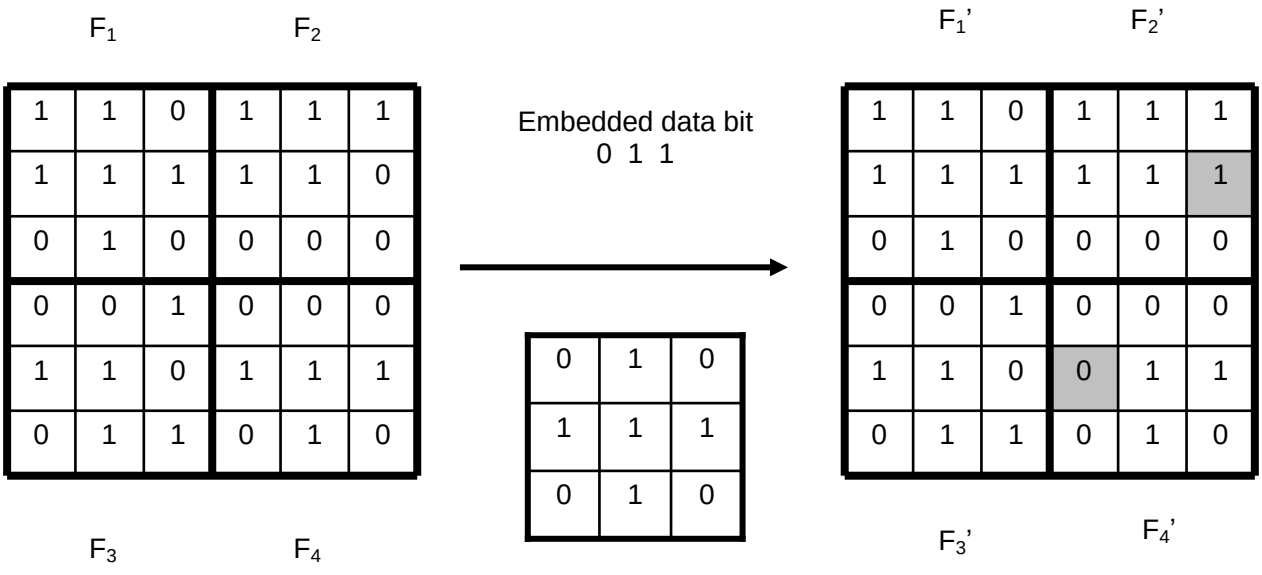



Рис. 12.9. Пример встраивания информации с использованием ключа.

На рис. 12.9 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.

$F_1$: sum($F_1 \Lambda K$)=5=sum(K), значит 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 в 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.

Учитывая перечисленные выше недостатки, был разработан еще один алгоритм, который является усложненной модификацией первых двух. Во-первых, в нем изображение уже разделяется не на блоки 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 1 бита информации нам необходим блок размером 7×7. Таким образом, для встраивания 1 символа необходимо 8 блоков или 49×8=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).

## 3. 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 12.10.

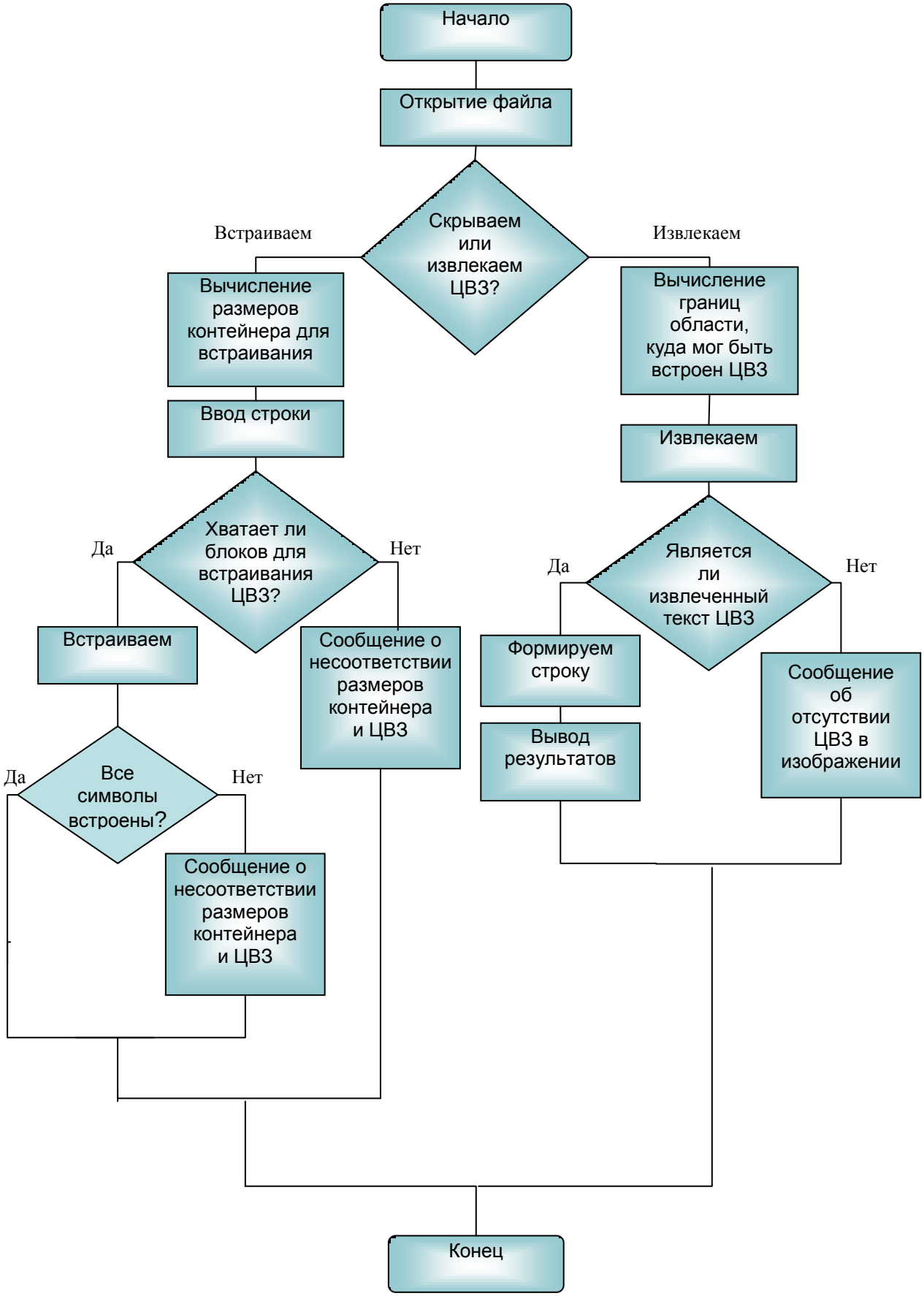



Рис. 12.10. Обобщенный алгоритм работы ПО, встраивающего ЦВЗ

### 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. 12.11.

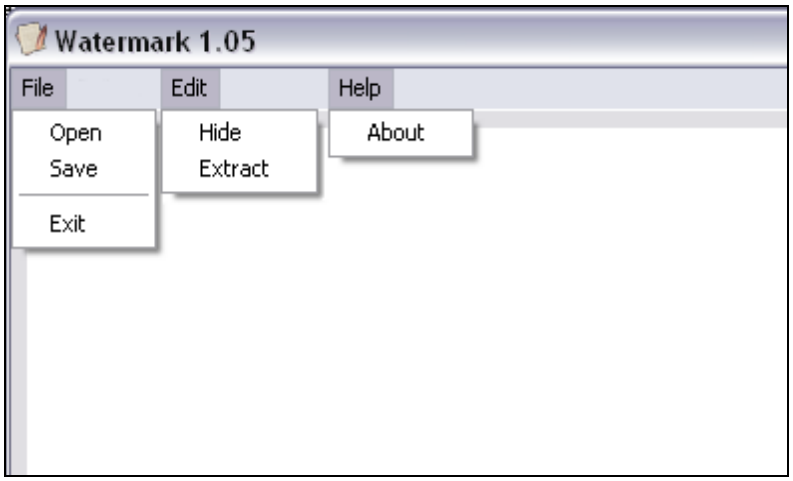



Рис. 12.11. Структура главного меню

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.

### 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.

В случае, если выбранный файл представляет собой не черно-белое изображение, пользователю выдается сообщение об ошибке, рис. 12.12.


Рис. 12.12. Сообщение о неподходящем формате открываемого файла

После открытия изображения, пользователь выбирает – извлекаться будет ЦВЗ, или встраиваться. Для встраивания, в текстовую строку приложения должен быть введен текст метки, рис 12.13.

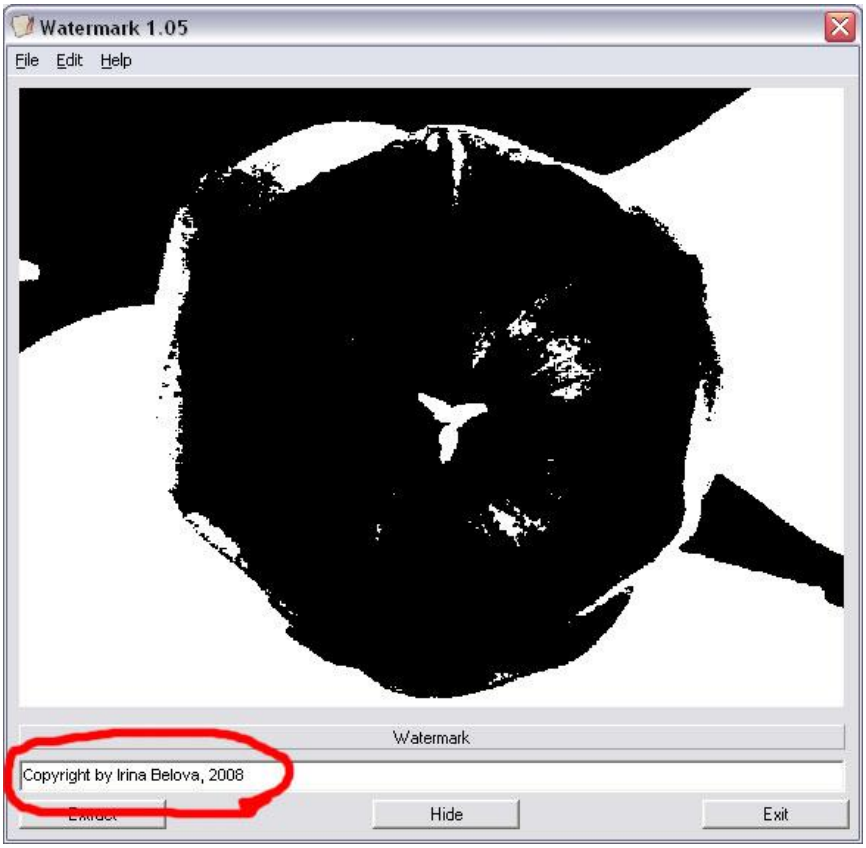



Рис. 12.13. Цифровой водяной знак

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. 12.14. 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. 12.14. 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.12.15, иначе в текстовой строке будет выведен текст ЦВЗ, рис. 12.16.


Рис. 12.15. Сообщение о том, что ЦВЗ не обнаружен

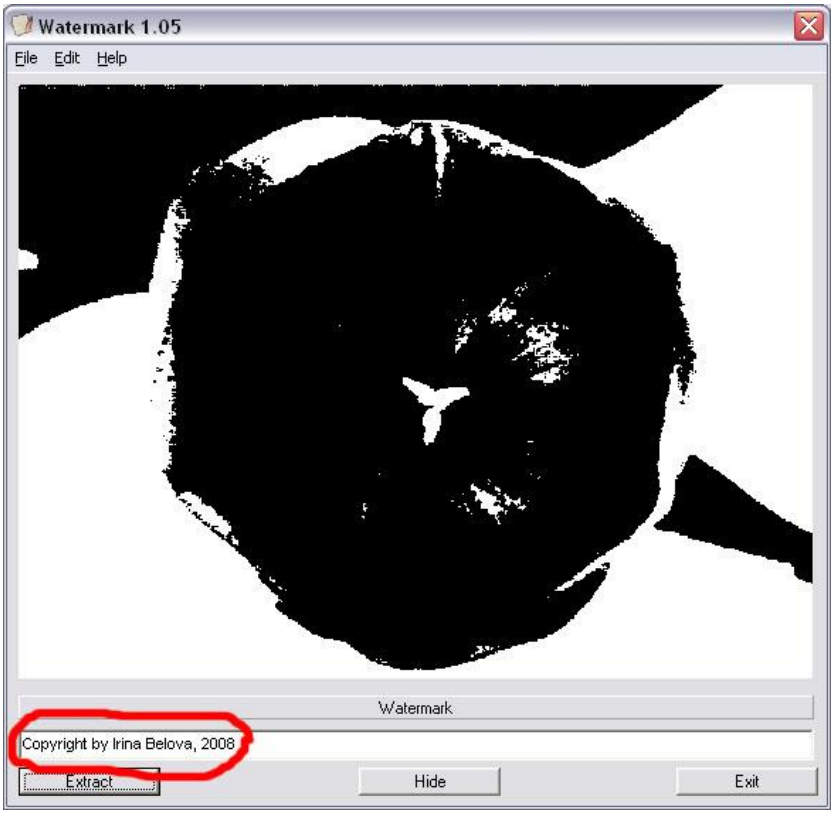
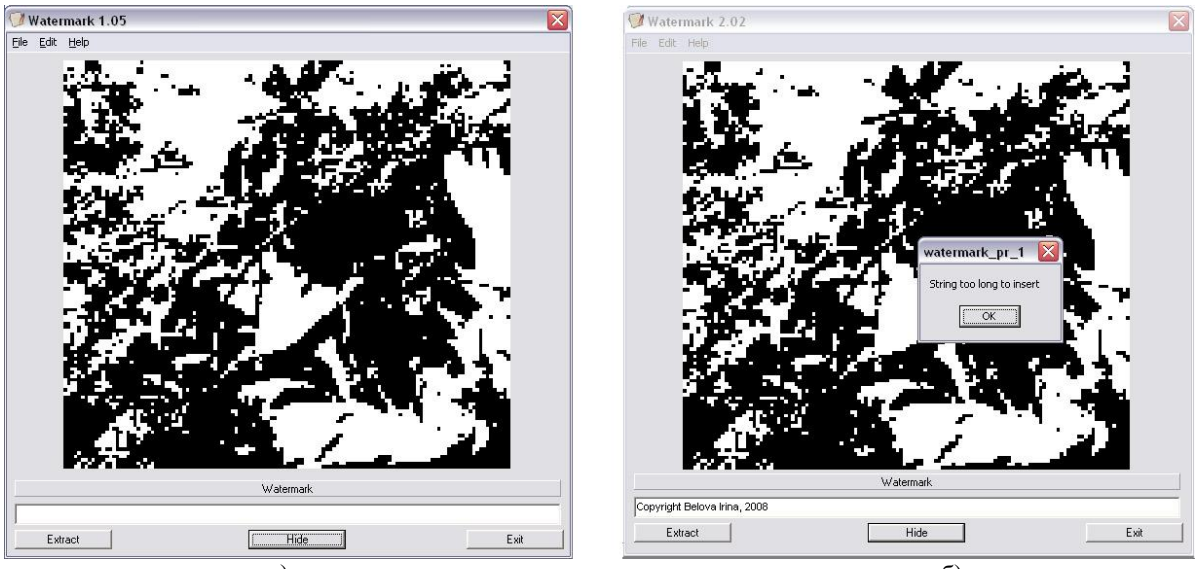



Рис. 12.16. Пример извлечения ЦВЗ

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. 12.17 и рис. 12.18.

Рис. 12.17. Пустой контейнер размером 128×117 пикселей




а) б)

Рис. 12.18. 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 не достаточно хорошие результаты по сохранению внешнего вида исходного изображения. Примеры работы программ для таких изображений представлены на рис.12.19 и рис.12.20. В изображение размером 400×300 пикселей встраивается ЦВЗ из 28 символов.

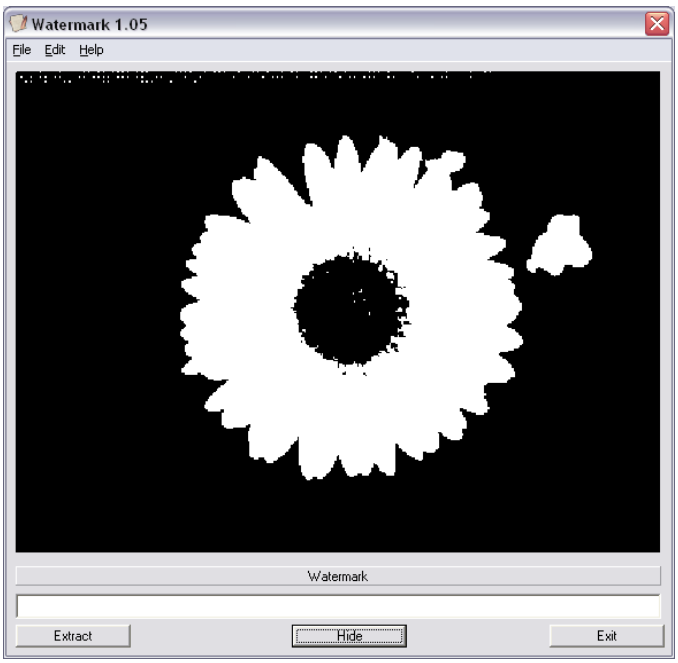



Рис. 12.19. Встраивание ЦВЗ в блоки изображения размером 3×3 пикселя

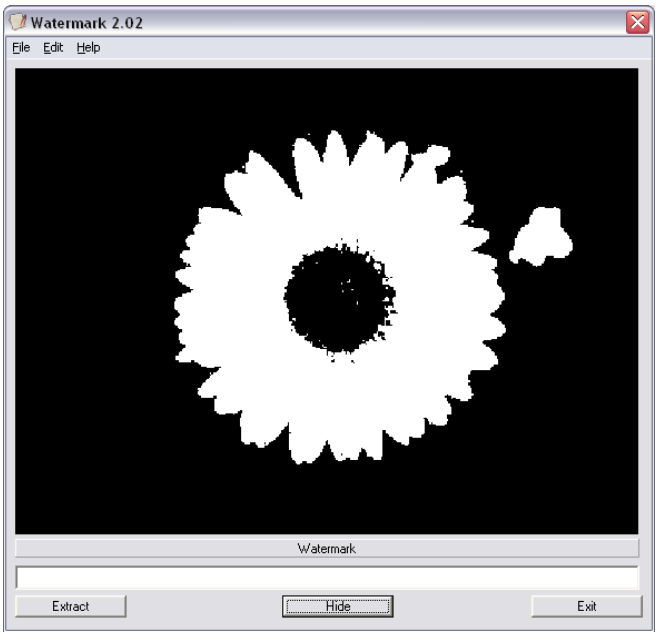



Рис. 12.20. Встраивание ЦВЗ в блоки изображения размером 7×7 пикселей

## 4. Порядок выполнения лабораторной работы

1. Запустите программу ***watermark_1.05***, предварительно установленную на ваш компьютер.

2. Выберете 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 12.13). 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. 12.14, 12.15). Сравните их качество, сделайте выводы об эффективности алгоритма извлечения ЦВЗ.
5. Помеченные файлы-контейнеры подвергните обработке:

   – с использованием алгоритмов архивации с потерями;

– с использованием 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.

**Требования к отчету**

Сохранить в отчете экранные формы, демонстрирующие процесс встраивания ЦВЗ в различные файлы-контейнеры. Сделать выводы об эффективности изученного метода защиты.

Включить в отчет о лабораторной работе ответы на контрольные вопросы, выбранные в соответствии с номером варианта из *приложения 1*.

*Приложение 1*

| **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 №13

# СТЕГОКОМПЛЕКСЫ, ДОПУСКАЮЩИЕ ИСПОЛЬЗОВАНИЕ АУДИО КОНТЕЙНЕРОВ, 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.***

## 1. Описание программного пакета Invisible Secrets-4

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 аудио данные, 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 рис.13.1 показан результат работы программы *Invisible Secrets-4* на тестовых контейнерах, при оцифровке 8 бит на отсчет.

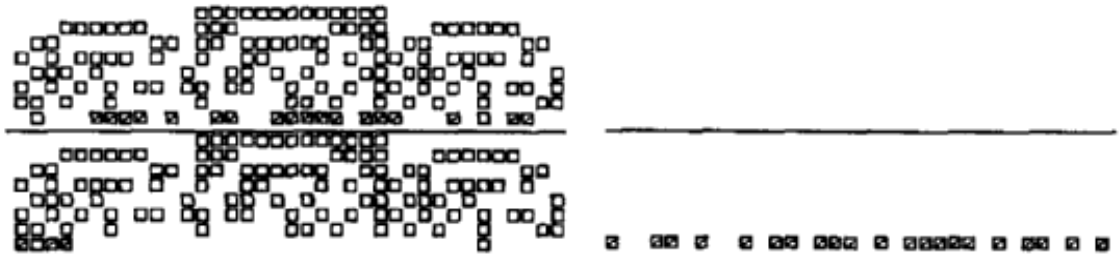


Рис. 13.1.Результат работы программы на тестовых контейнерах

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 рис.13.2 приведен вид главного меню программы *Invisible Secrets -4.*


Рис. 13.2. 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.

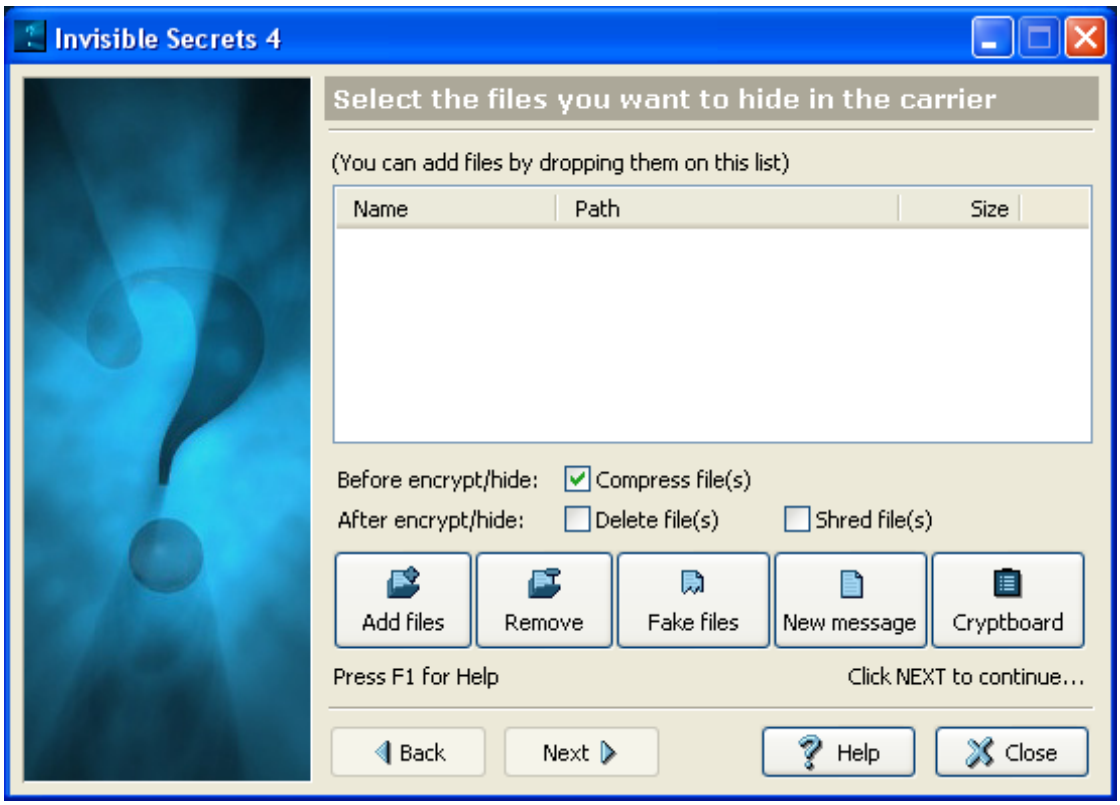



Рис. 13.3. Выбор файла, который необходимо встроить или зашифровать, и задание соответствующих параметров

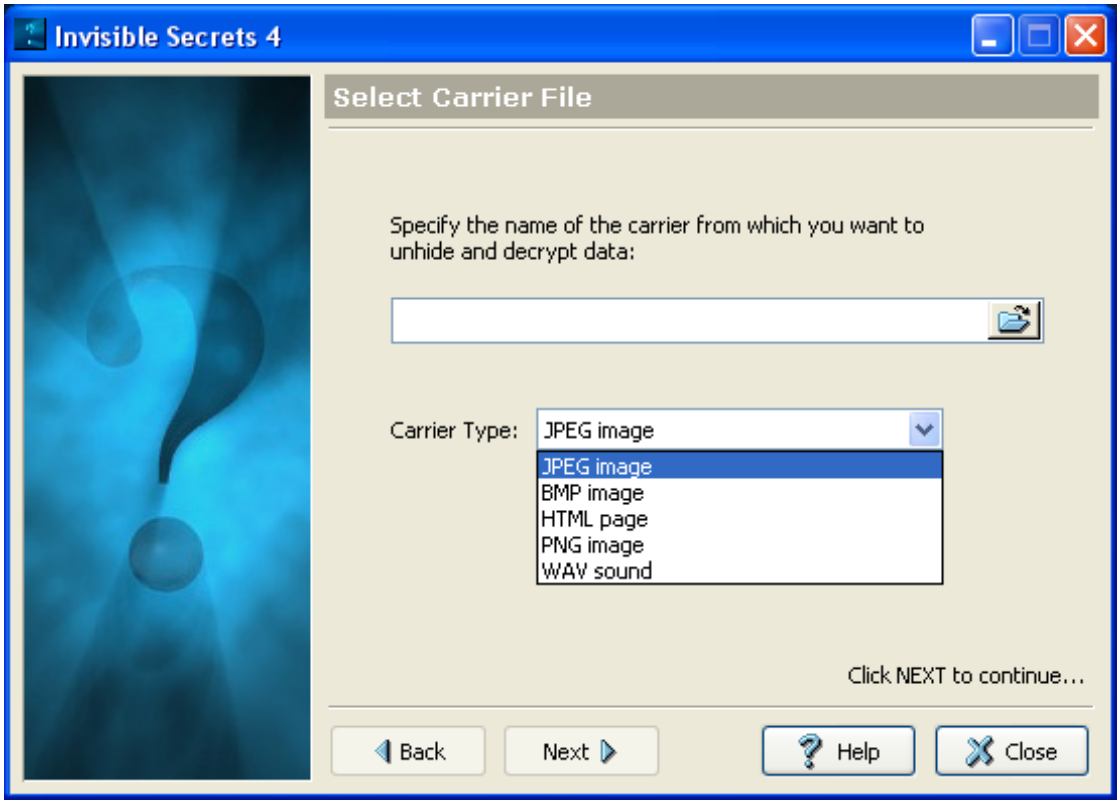



Рис. 13.4. Выбор файла для извлечения из файла-контейнера или расшифрования

**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, RC4, Cast128, Diamond 2, Sapphire II.* На рис. 13.5 приведена панель шифрования файлов, а на рис. 13.6 панель расшифрования файлов.

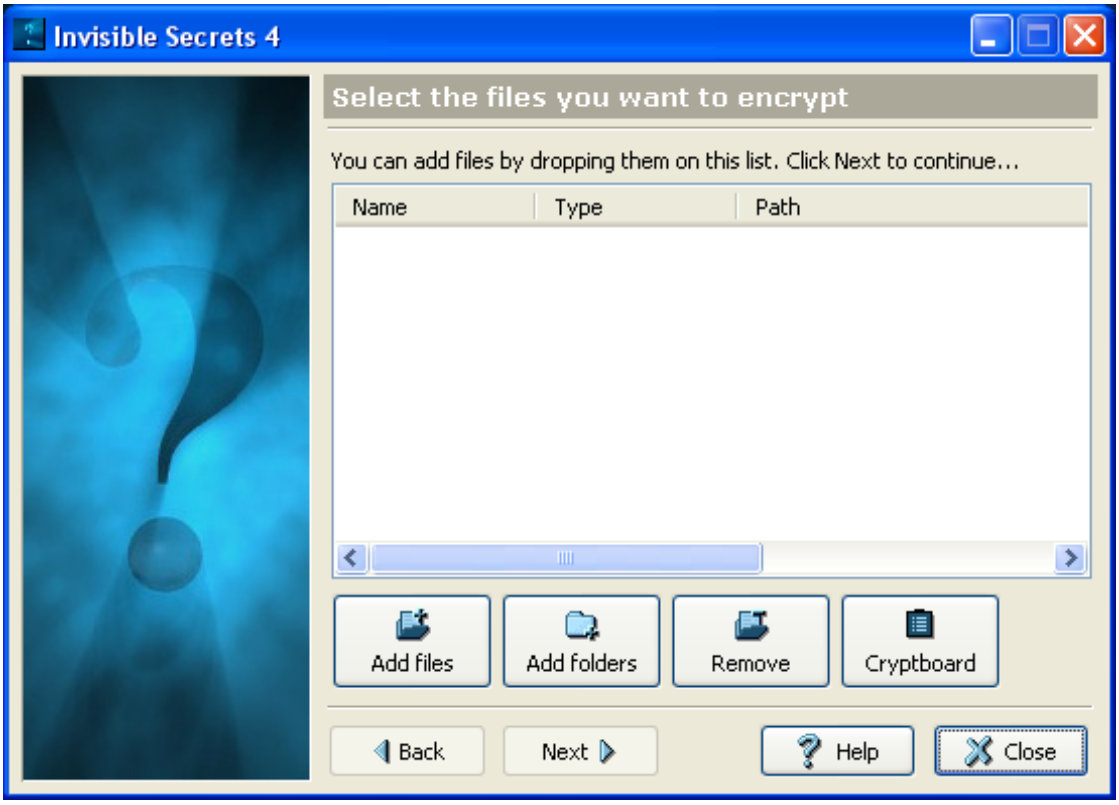



Рис. 13.5. Панель шифрования файлов

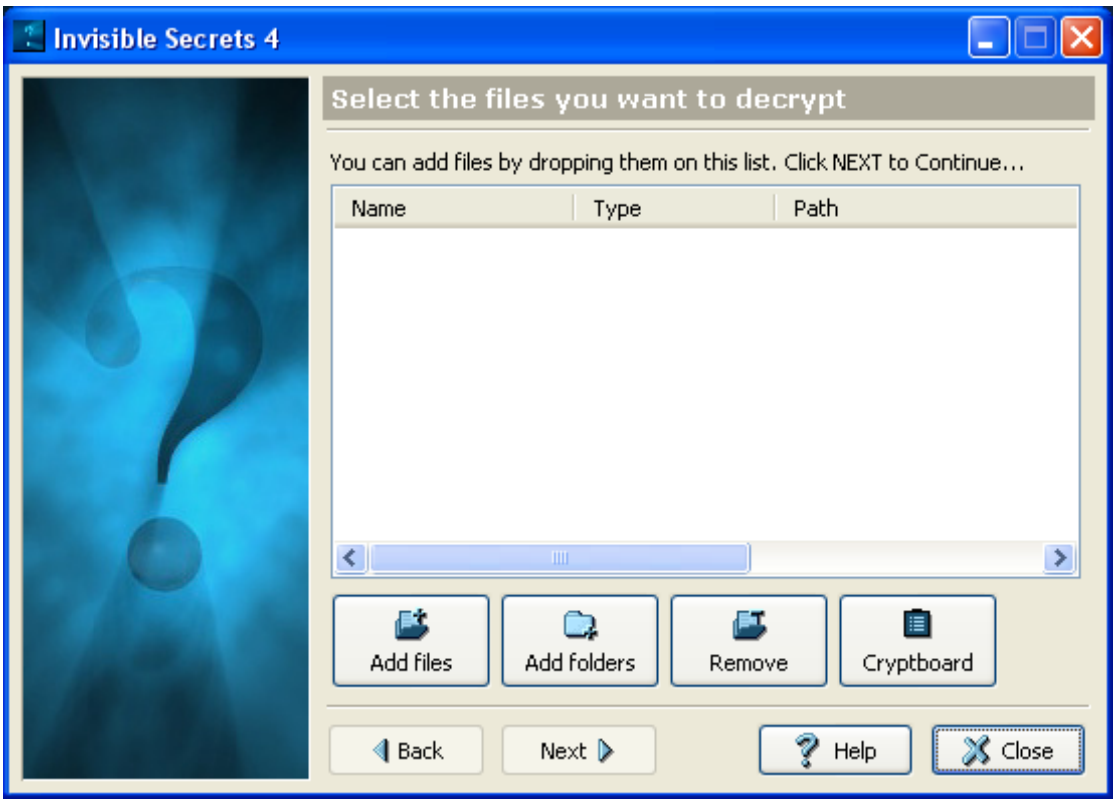



Рис. 13.6. Панель расшифрования файлов

**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 рис.13.7 приводится панель программы *Invisible Secrets-4* для создания самораспаковывающегося пакета для его отправки по *e-mail.*

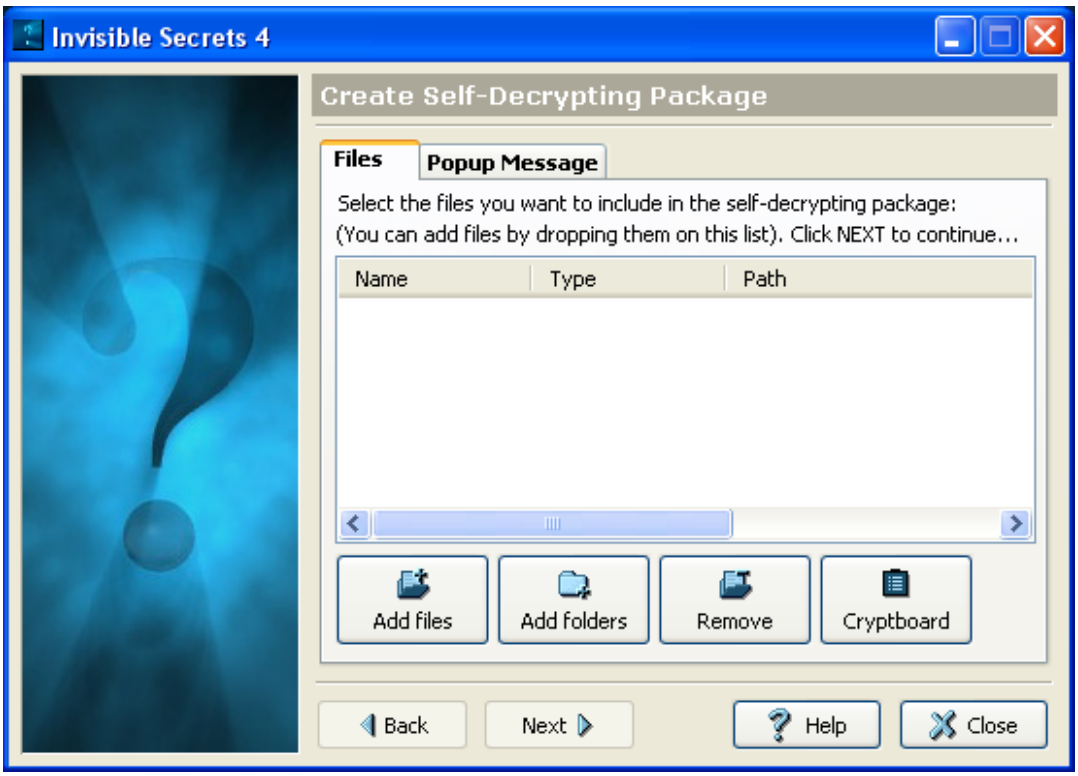



Рис. 13.7. Создание самораспаковывающегося пакета для его отправки по *e-mail*

### 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. 13.8, а для уничтожения ссылок, на рис. 13.9.

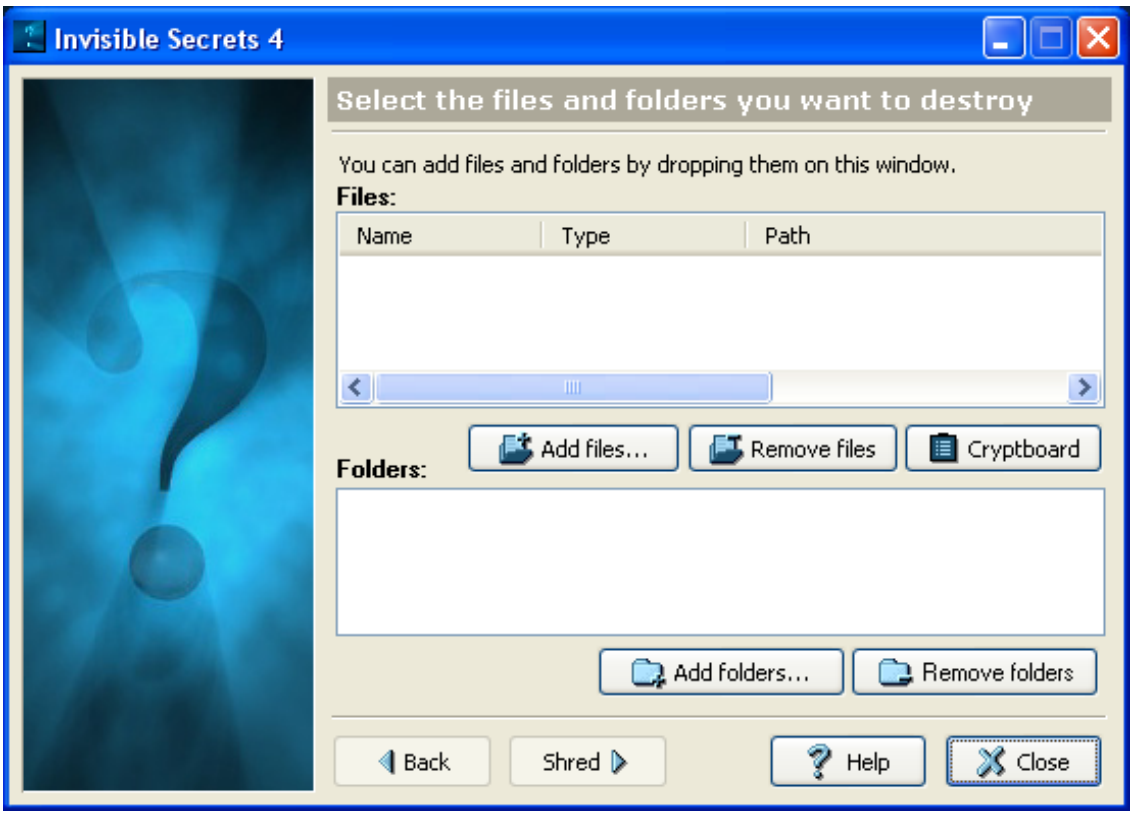



Рис.13.8. Выбор папки или файла, которые  необходимо уничтожить

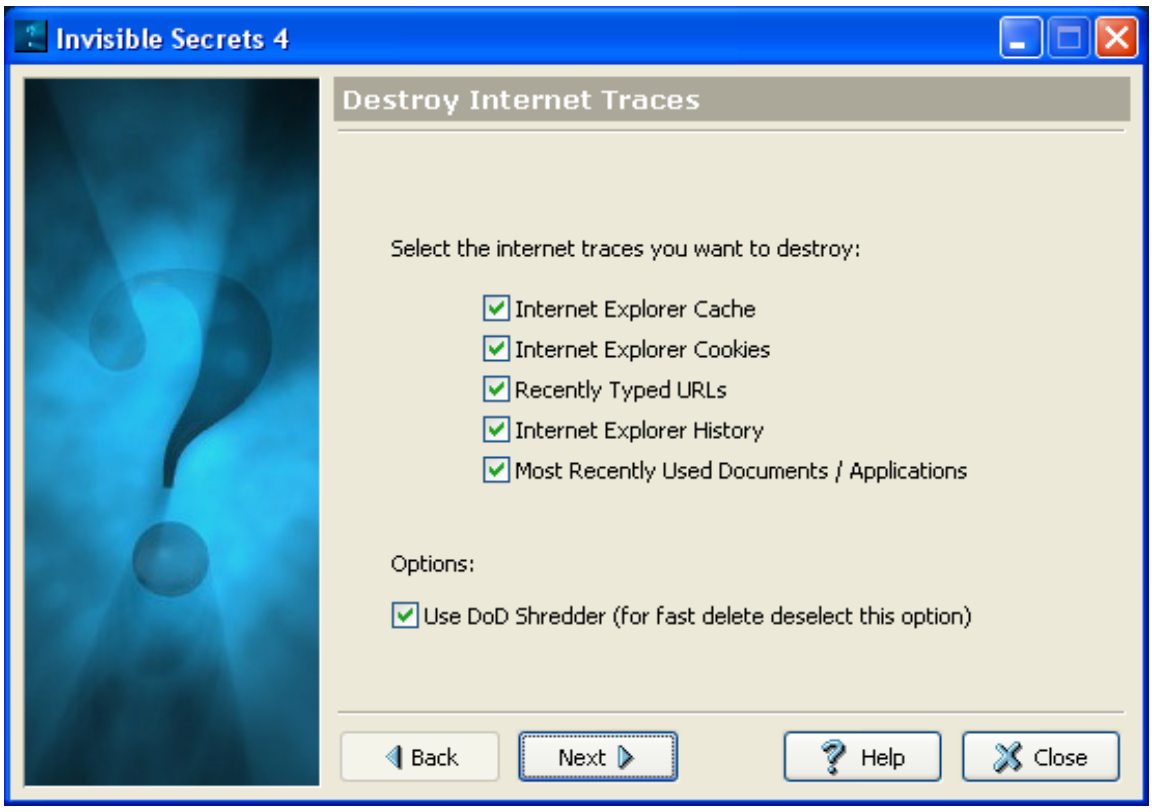



Рис.13.9. Уничтожение ссылок

**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*

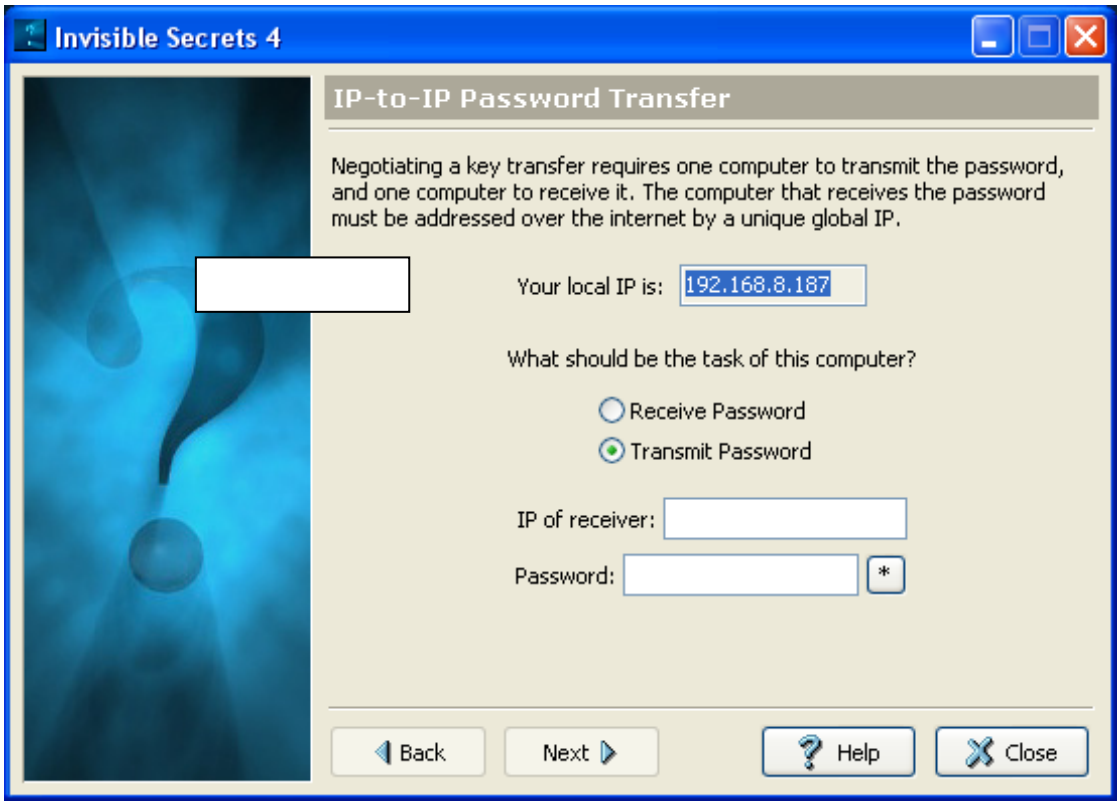



Рис.13.10. Безопасная передача пароля

## 2. Порядок выполнения лабораторной работы

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 *приложения 1*.

*Приложение 1*

| **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 аудио файлами 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. Бабаш А.В., Шанкин Г.П. Криптография. / Под редакцией В.П.Шерстюка, Э.А. Применко. – М.: СОЛОН-Р, 2002. – 512 с.
2. Башлы П.Н., Бабаш А.В., Баранова Е.К. Информационная безопасность: учебно-практическое пособие. – М.: Изд. Центр ЕАОИ, 2010. – 376 с.
3. Грибунин В.Г., Оков И.Н., Туринцев И.В. Цифровая стеганография. – М.: СОЛОН-Пресс, 2002. – 272 с.
4. Конахович Г.Ф. Компьютерная стеганография: теория и практика. – М.: МК-Пресс, 2006. – 221 с..
5. Митекин В. А. Модифицированные методы статистического стегоанализа бинарных и полутоновых изображений. // Компьютерная оптика, 2005, №28, С. 145-152.
6. Мотуз О. В. Защита авторских прав с помощью цифровых «водяных знаков» // Защита информации. Конфидент. 1998. № 6, С. 66-70.
7. Fu M. S., Au O. C. Data hiding by smart pair togglingfor halftone images // Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, Istanbul, Turkey, 2000. Vol. 4. P. 2318-2321.