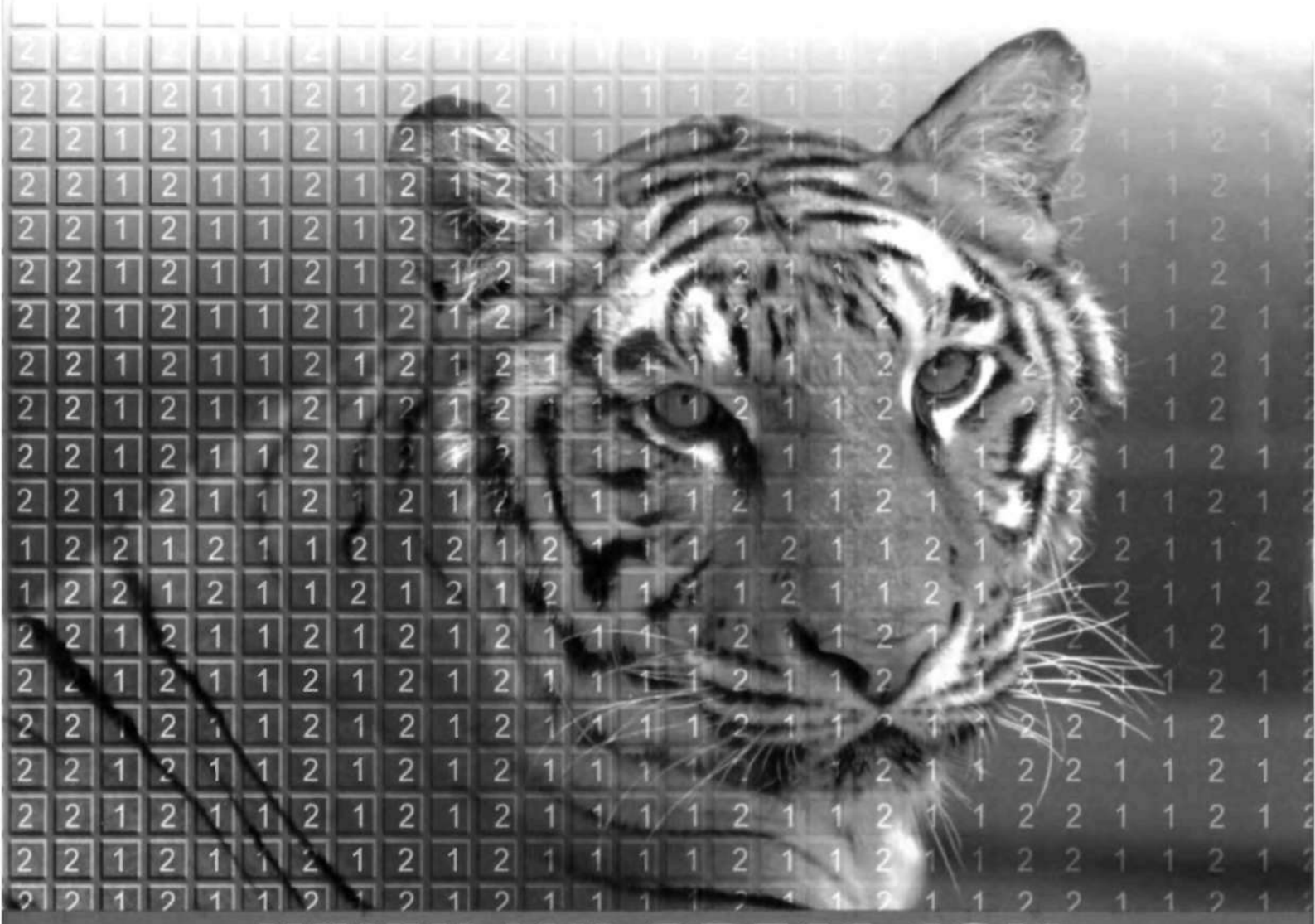
Кафедра телекоммуникационных систем

Г.Ф. Конахович А.Ю. Пузыренко

# КОМПЬЮТЕРНАЯ СТЕГАНОГРАФИЯ

## ТЕОРИЯ И ПРАКТИКА



WWW.MK-PRESS.COM

Г. Ф. Конахович, А. Ю. Пузыренко

# Компьютерная стеганография

## ТЕОРИЯ И ПРАКТИКА

**"МК-Пресс"**
**Киев, 2006**

ББК 32.811.4
К 338
УДК 519.688

**Конахович Г. Ф., Пузыренко А. Ю.**
К338 Компьютерная стеганография. Теория и практика.— К.: "МК-Пресс", 2006. — 288 с, ил.
ISBN 966-8806-06-9

Эта книга — одно из первых изданий в области стеганографии. В ней изложены теоретические и практические основы компьютерной стеганографии. Представлены особенности использования современной системы символьной математики MathCAD v.12 в целях стеганографической защиты информации. Рассмотрены примеры практической реализации скрытия данных в неподвижных изображениях, аудиосигналах и текстах.

Системно изложены проблемы надежности и стойкости произвольной стеганографической системы по отношению к различным видам атак, а также оценки пропускной способности канала скрытого обмена данными. Представлены результаты существующих информационно-теоретических исследований проблемы информационного скрытия в случае активного противодействия нарушителя. Также рассмотрены известные стеганографические методы, направленные на скрытие конфиденциальных данных в компьютерных файлах графического, звукового и текстового форматов.

**ББК 32.811.4**

**Конахович Георгий Филимонович**
**Пузыренко Александр Юрьевич**

# Компьютерная стеганография

## ТЕОРИЯ И ПРАКТИКА

*Главный редактор: Ю. А. Шпак*

Подписано в печать 20.12.2005. Формат 70 χ 100 1/16.
Бумага газетная. Печать офсетная. Усл. печ. л. 23,3. Уч.-изд. л. 18,7.
Тираж 1000 экз. Заказ №_____

ЧП Савченко Л.А., Украина, г.Киев, тел./ф.: (044) 517-73-77; e-mail: info@mk-press.com.
Свидетельство о внесении субъекта издательского дела в Государственный реестр издателей, производителей и распространителей издательской продукции: серия ДК №51582 от 28.11.2003г.

Отпечатано в ЧП "КОРВИН ПРЕСС". г.Киев, ул. Пшеничная, 2

ISBN 966-8806-06-9

# Содержание

# Перечень условных сокращений

**АКФ** автокорреляционная функция
**АС** автоматизированная система
**БПФ** быстрое преобразование Фурье
**БЧХ** код Боуза-Чоудхури-Хоквенгема
**ВЧ** высокочастотный (сигнал)
**ГПСП** генератор псевдослучайной перестановки
**ГПСФ** генератор псевдослучайной функции
**ДКП** дискретное косинусное преобразование
**ДПФ** дискретное преооразование Фурье
**ЗСЧ** зрительная система человека
**ИКМ** импульсно-кодовая модуляция
**КОС** канал открытой связи
**КСС** канал скрытой связи
**ЛРСОС** линейный регистр сдвига с обратной связью
**КПСД** канал передачи скрытых данных
**КС** компьютерная стеганография
**МСЭ** международный союз электросвязи
**НЗБ** наименьший значащий бит
**НЧ** низкочастотный (сигнал)
**ПО** программное обеспечение
**ПС** пропускная способность
**ПСП** псевдослучайная последовательность
**ПСЧ** псевдослучайное число
**ПКЛ** преобразование Карунена-Лоева
**РСПП** расширение спектра (сигнала) прямой последовательностью
**СПС** скрытая пропускная способность
**ССЧ** слуховая система человека
**ЦВЗ** цифровой водяной знак
**ЦОС** цифровая обработка сигналов
**ЦС** цифровая стеганография
**ШПМ** широкополосный метод
**ASCII** American Standard Code for Information Interchange — американский стандартный код для обмена информацией. Набор из 128 кодов символов для машинного представления прописных и строчных букв латинского алфавита, чисел, разделительных знаков и специальных символов, каждому из которых соответствует конкретное 7-битное двоичное число. Первые 32 символа данного кода являются управляющими (такими как "перевод строки", "возврат каретки") и служат для управления печатью и передачи данных. Они не могут быть выведенными в текстовом виде. Восьмой бит при передаче данных может использоваться для контроля четности или для расширенного набора символов ASCII *{extended ASCII* \ содержащего буквы разных языков и графические символы.
**BMP** BitMaP— битовое (растровое) отображение графического объекта. Используется для представления изображений. Стандартный формат графических файлов, который предусматривает 4, 8 и 24 бита квантования на один пиксель.

CR Carriage Return — служебный ASCII-код, который обозначает операцию возврата курсора (каретки) — его перевода к левому краю листа при выводе текста на символьное устройство.

**DCT** Discrete Cosine Transform — дискретное косинусное преобразование — математическое преобразование, используемое в алгоритмах компрессии изображений (например, в JPEG).

FDCT Forward DCT — прямое дискретное косинусное преобразование.

**GIF** Graphics Interchange Format — формат обмена графическими данными, разработанный информационной службой CompuServe в 1987 г. для эффективной пересылки графики (GIF87a). Широко используется для сохранения простых растровых изображений, содержащих большие поля одного цвета.

**ГОСТ** Inverse DCT — обратное дискретное косинусное преобразование.

JPEG стандарт сжатия с потерями для неподвижных полноцветных видеоизображений на основе алгоритма дискретного косинусного преобразования с коэффициентом компрессии данных более 25:1. Разработан группой экспертов по машинной обработке фотографических изображений (Joint Photographic Experts Group). Непересекающиеся блоки изображения размерами 8x8 пикселей обрабатываются с применением целочисленной арифметики.

**LF** Line Feed — служебный ASCII-код, который вызывает перевод курсора на экране в ту же самую колонку на одну строку ниже.

**LFSR** Linear Feedback Shift Register — линейный регистр сдвига с обратной связью

**LSB** Least Significant Bit — младший значащий бит (разряд) двоичного числа.

**MSB** Most Significant Bit — старший значащий бит (разряд) двоичного числа.

**PCM** Pulse Code Modulation — импульсно-кодовая модуляция.

**RGB** Red-Green-Blue — "красный-зеленый-синий" — основная палитра, используемая в программировании и компьютерной графике.

**VPN** Virtual Private Network — виртуальная частная сеть — подсеть корпоративной сети, обеспечивающая безопасный вход в нее удаленных пользователей. Используется для безопасной пересылки по сети Internet конфиденциальных данных за счет инкапсуляции (туннелирования) IP-пакетов внутри других пакетов, которые затем маршрутизуются.

# Вступление

Информация является одним из ценнейших предметов современной жизни. Получение доступа к ней с появлением глобальных компьютерных сетей стало невероятно простым. В то же время, легкость и скорость такого доступа значительно повысили и угрозу нарушения безопасности данных при отсутствии мер относительно их защиты, а именно, — угрозу неавторизированного доступа к информации.

Задача надежной защиты авторских прав, прав интеллектуальной собственности или конфиденциальных данных (которые в большинстве случаев имеют цифровой формат) от несанкционированного доступа является одной из старейших и нерешенных на сегодня проблем. В связи с интенсивным развитием и распространением технологий, которые позволяют с помощью компьютера интегрировать, обрабатывать и синхронно воспроизводить различные типы сигналов (так называемые мультимедийные технологии), вопрос защиты информации, представленной в цифровом виде, является чрезвычайно актуальным.

Преимущества представления и передачи данных в цифровом виде (легкость восстановления, высокая потенциальная помехоустойчивость, перспективы использования универсальных аппаратных и программных решений) могут быть перечеркнуты с легкостью, с которой возможны их похищение и модификация. Поэтому во всем мире назрел вопрос разработки методов (мер) по защите информации организационного, методологического и технического характера, среди них — методы криптографии и стеганографии.

*Криптографическая* (с греческого *χρυπτός* — "тайный", *γράφω* — "пишу") защита информации (система изменения последней с целью сделать ее непонятной для непосвященных, сокрытие содержания сообщений за счет их шифрования) не снимает упомянутую выше проблему полностью, поскольку наличие шифрованного сообщения само по себе привлекает внимание, и злоумышленник, завладев криптографически защищенным файлом, сразу понимает о размещении в нем секретной информации и переводит всю суммарную мощь своей компьютерной сети на дешифрование данных.

Скрытие же самого факта существования секретных данных при их передаче, хранении или обработке является задачей *стеганографии* (от греческого *στεγανός* — "скрытый") — науки, которая изучает способы и методы скрытия конфиденциальных сведений. Задача извлечения информации при этом отступает на второй план и решается в большинстве случаев стандартными криптографическими методами.

Иначе говоря, под скрытием существования информации подразумевается не только невозможность обнаружения в перехваченном сообщении наличия иного (скрытого) сообщения, но и вообще сделать невозможным возникновение любых подозрений на этот счет, поскольку в последнем случае проблема информационной безопасности возвращается к стойкости криптографического кода. Таким образом, занимая свою нишу в обеспечении безопасности, стеганография не заменяет, а дополняет криптографию [1].

Стеганографирование осуществляется различными способами. Общей же чертой таких способов является то, что скрываемое сообщение встраивается в некий непривлекающий внимание объект, который затем открыто транспортируется (пересылается) адресату.

Исторически направление стеганографического скрытия информации было первым [2], но со временем во многом было вытеснено криптографией. Интерес к стеганографии возродился в последнее десятилетие и был вызван широким распро-

странением технологий мультимедиа (что вполне закономерно, принимая во внимание указанные выше проблемы, связанные с защитой информации). Не менее важным стало появление новых типов каналов передачи информации, что в совокупности с первым фактором дало новый импульс развитию и усовершенствованию стеганографии, способствовало возникновению новых стеганографических методов, в основу которых были положены особенности представления информации в компьютерных файлах, вычислительных сетях и т.д. Это, в свою очередь, дает возможность говорить о становлении нового направления в сфере защиты информации: ***компьютерной стеганографии*** **(КС) [3-5,19].**

С 1996 г. проводятся международные симпозиумы по проблемам скрытия данных (Information Workshop on Information Hiding). Первая конференция, посвященная стеганографии, состоялась в июле 2002 г. На сегодняшний день стеганография является наукой, которая быстро и динамично развивается, используя при этом методы и достижения криптографии, цифровой обработки сигналов, теории связи и информации.

Методы стеганографии позволяют не только скрыто передавать данные (так называемая *классическая стеганография),* но и успешно решать задачи помехоустойчивой аутентификации, защиты информации от несанкционированного копирования, отслеживания распространения информации сетями связи, поиска информации в мультимедийных базах данных и т.д. Эти обстоятельства позволяют в рамках традиционно существующих информационных потоков или информационной среды решать некоторые важные вопросы защиты информации ряда прикладных отраслей.

Существует два ключевых направления использования КС: *связанное* с цифровой обработкой сигналов (ЦОС) и не связанное. В первом случае секретные сообщения встраиваются в цифровые данные, которые, как правило, имеют аналоговую природу (речь, изображение, аудио- и видеозаписи) [1,3-5]. Во втором — конфиденциальная информация размещается в заголовках файлов или пакетов данных (это направление не нашло широкого применения из-за относительной легкости извлечения и/или уничтожения скрытой информации). Подавляющее большинство текущих исследований в сфере стеганографии так или иначе связано именно с ЦОС. что позволяет говорить о *цифровой стеганографии* (ЦС) [5,19].

Можно выделить по крайней мере две причины популярности в наше время исследований в сфере стеганографии: ограничение на использование криптографических средств в ряде стран мира и возникновение проблемы защиты прав собственности на информацию, представленную в цифровом виде.

Первая причина вызвала большое количество исследований в духе классической стеганографии (то есть, скрытие собственно факта передачи), а вторая — не менее многочисленные работы в сфере так называемых *цифровых водяных знаков* (ЦВЗ) — специальных меток, скрыто встроенных в изображение (или другие цифровые данные) с тем, чтобы иметь возможность контролировать его использование.

Скрытие информации только на основе факта неизвестности злоумышленнику метода или методов, заложенных в основу скрытия, на сегодняшний день является малоэффективным. Еще в 1883 году фламандский криптограф А. Керхгофс (A. Kerckhoffs) указывал на тот факт, что система защиты информации должна выполнять возложенные на нее функции даже при полной информированности противника о ее структуре и алгоритме функционирования [6].

Вся секретность системы защиты передаваемых сообщений должна содержаться в *ключе* — фрагменте информации, предварительно (как правило) разделенном между адресатами. Несмотря на то, что этот принцип известен уже более 100 лет, до

сих пор встречаются разработки, которые ими пренебрегают. Очевидно, что они не могут использоваться с серьезной целью.

В основе многих подходов к решению задач стеганографии лежит общая с криптографией методическая база, которую заложил еще в средине прошлого века К. Шеннон (CE. Shannon) [45,60]. Однако и до сих пор теоретические основы стеганографии остаются практически непроработанными.

Принимая во внимание вышесказанное, можно сделать вывод о том, что на сегодняшний день актуальна научно-техническая проблема усовершенствования алгоритмов и методов проведения стеганографического скрытия конфиденциальных данных или защиты авторских прав на определенную информацию.

Сегодня нет недостатка в стеганографических программах как начального, так и профессионального уровня (S-Tools, Steganos Security Suite, bmpPacker и др.), однако защищенность их кода (особенно это относится к программам профессионального уровня) не позволяет проследить методы, положенные в основу алгоритмов их действия. Размещенные же в Internet-ресурсах многочисленные тексты программ из-за своей низкой информативности мало чем помогают, так как компиляция предложенных текстов возвращает исполняемую программу, алгоритм которой крайне трудно проследить, поскольку последняя выдает уже готовый результат — заполненный стеганоконтейнер, — и практически не существует возможности заранее установить достаточность уровня скрытия конфиденциальной информации в этом контейнере.

Таким образом, совершенно очевидна нехватка именно *программ начального уровня,* которые бы наглядно, шаг за шагом демонстрировали весь процесс стеганографического преобразования, что можно было бы использовать в учебном процессе при подготовке специалистов в сфере защиты информации.

Состояние затронутого вопроса в сфере стеганографии характеризуется следующими основными достижениями. Вопросы стеганографического скрытия секретных сведений, включая построение эффективных алгоритмов скрытия, в свое время рассматривали в своих работах Симмонс (G.J. Simmons), Фридрих (J. Fridrich), Андерсон (R.J. Anderson), Бендер (W. Bender), Моримото (N. Morimoto), Качин (C. Cachin), Питас (I. Pitas) и другие [7-9, 13-16]. Результаты исследования стеганографических алгоритмов на устойчивость приводят в своих работах Фридрих (J. Fridrich), Попа (R. Popa), Джонсон (N.F. Johnson), Волошиновский (S. Voloshynovskiy) [9, 17, 18, 20, 40, 41]. Также необходимо отметить работы Пфицманна (B. Pfitzmann), Шнайера (B. Schneier) и Кравера (S. Craver) по вопросам согласования терминологии и формирования основных стеганографических протоколов [10-12].

Длительное время в отечественной литературе и литературе стран СНГ стеганографии было посвящено лишь несколько обзорных журнальных статей [1,4, 22-24, 48]. Кроме того, заслуживает внимания работа [5] таких авторов как Грибунин В. Г., Оков И. М., Туринцев И. В.

В последнее время отсутствие научной литературы указанной тематики отечественных авторов в определенной мере ликвидировано такими специалистами как В. А. Хорошко, А. Д. Азаров, М. Е. Шелест и др. [3], заслугой которых является едва ли не первая попытка системного изложения стеганографических методов, обобщение новейших результатов исследований в сфере компьютерной стеганографии.

Целью данной работы является изложение теоретических, и, что не менее важно, практических основ компьютерной стеганографии. Рассмотрены особенности и перспективы использования современной системы символьной математики MathCADv.12 в целях стеганографической защиты информации. Выполнен анализ спе-

циализированных литературных источников и ресурсов сети Internet относительно перспективных направлений, в которых возможно использование стеганографии в качестве инструмента защиты информации в автоматизированных системах обработки данных.

Путем исследования известных публикаций отечественных и зарубежных авторов осуществлено системное изложение проблем надежности и стойкости произвольной стеганографической системы по отношению к видам осуществляемых на нее атак, а также оценки пропускной способности канала скрытого обмена данными, которым, по сути, является стеганосистема. Представлены результаты существующих информационно-теоретических исследований проблемы информационного скрытия в случае активного противодействия нарушителя. Кроме того, в книге системно изложены известные стеганографические методы, направленные на скрытие конфиденциальных данных в компьютерных файлах графического, звукового и текстового форматов.

Представлены примеры программных комплексов для демонстрации принципов, положенных в основу методов стеганографического скрытия информации в пространственной (временной) или частотной областях используемого контейнера (неподвижного изображения, аудиосигнала или текстового документа).

Применение при компьютерном моделировании универсальной математической системы MathCAD v.12 позволяет использовать мощные средства реализации численных методов расчета и математического моделирования совместно с возможностями выполнения операций символьной математики [25, 26]. Стороны, осуществляющие скрытый обмен данными, практически избавляются от необходимости программировать собственно решение задачи — на них лишь возлагается корректное описание алгоритма решения на входном языке MathCAD, который является языком очень высокого уровня.

Указанное является существенным преимуществом по сравнению с существующими на сегодня программами, написанными при помощи таких низкоуровневых языков как C/C++, Basic и визуальных интерфейсов на их основе. Последние, хотя и отличаются достаточно высоким уровнем гибкости с точки зрения возможностей реализации тех или иных методов стеганографии, однако характеризуются несравнимо длительным внесением изменений в уже написанную и откомпилированную программу. Время, затраченное не внесение модификаций, становится особенно важным в случае многоэтапных исследований, имеющих место при использовании программ в учебном процессе.

Благодаря своей наглядности и возможности быстрой модификации программных модулей, разработанные комплексы отвечают требованиям, которые предъявляются к программам, используемым в учебных целях. Подход объединения теоретического изложения материала с демонстрацией его практического использования позволяет избавиться от абстрактности формулировок, принятых в специализированной и справочной литературе по информационной безопасности, и способствует развитию здорового интереса к практическим аспектам решения научно-технических задач по защите информации. Книга может использоваться в качестве краткого справочного пособия по вопросам компьютерной стеганографии при использовании современных компьютерно-математических систем.

Задача данной книги — разработка программных комплексов для демонстрации принципов, заложенных в основу распространенных на сегодня методов стеганографической защиты с возможностью вычисления основных показателей искажения контейнера при встраивании в него скрываемых данных.

Эта задача решается предварительной проработкой следующих вопросов:

- рассмотрение особенностей построения стеганографических систем и основных типов атак на указанные системы;
- анализ современных исследований и публикаций по вопросам существующих методов стеганографии и мер повышения их пропускной способности и стойкости к стеганоанализу;
- формулирование практических рекомендаций относительно встраивания данных.

Книга предназначена для специалистов, которые работают в области зашиты информации и заинтересованы в эффективном использовании возможностей современных вычислительных систем, а также для студентов и преподавателей ВУЗов, специализирующихся в области информационной безопасности.

# Глава 1

# Место стеганографических систем в сфере информационной безопасности

## 1.1. Атаки на информацию, обрабатываемую в автоматизированных системах

Системообразующими факторами во всех сферах национальной безопасности являются уровень развития и безопасность информационного пространства. Они активно влияют на состояние политической, экономической, оборонной и других составляющих национальной безопасности страны.

*Информационная безопасность,* защита которой, наряду с суверенитетом, территориальной целостностью и экономической безопасностью, является важнейшей функцией государства, достигается путем разработки современного законодательства, внедрения современных безопасных информационных технологий, построением функционально полной национальной инфраструктуры, формированием и развитием информационных отношений и т.п.

Так, закон Украины "О защите информации в информационно-телекоммуникационных системах" [33] предлагает следующее определение понятия *'защита информации"* в информационных, телекоммуникационных и информационно-телекоммуникационных системах: это деятельность, направленная на предотвращение несанкционированных действий относительно информации в указанных системах. Отдельным видом защиты информации является *техническая защита,* реализованная с помощью инженерно-технических мероприятий и/или программных и технических средств, которые делают невозможными утечку, уничтожение и блокирование (невозможность доступа) информации, нарушение целостности (изменение содержимого) и режима доступа к информации. *Доступом к информации* является получение пользователем (физическим или юридическим лицом) возможности обрабатывать информацию в системе. Действия, которые производятся с нарушением порядка доступа к информации, установленного согласно законодательства, являются ***несанкционированными.***

Неоднозначным, по мнению специалистов, является трактовка понятия "атака на информацию", поскольку последняя, особенно в электронном виде, может быть представлена сотнями разнообразных видов. Информацией можно считать и отдельный файл, и базу данных, и лишь одну запись в ней, и целый программный комплекс. На сегодняшний день все эти объекты подвергаются или могут быть подвергнуты атакам со стороны некоторого лица (группы лиц) — *нарушителя.*

При хранении, поддержке и предоставлении доступа к любому информационному объекту его владелец (или уполномоченное им лицо) накладывает (явно или

самоочевидно) набор правил по работе с ним. Преднамеренное их нарушение и классифицируется как ***атака на информацию*** [34-37].

С массовой компьютеризацией всех сфер деятельности человека объем информации, хранимой в электронном виде, возрос в тысячи раз. Это, в свою очередь, значительно повысило риск *утечки* ***информации,*** в результате чего последняя становится известной (доступной) субъектам, не имеющим права доступа к ней. С появлением компьютерных сетей даже отсутствие физического доступа к компьютеру перестало быть гарантией сохранности информации.

Рассмотрим возможные последствия атак на информацию, в первую очередь — ***экономические потери:***

- раскрытие коммерческой информации может привести к значительным прямым убыткам;
- сведения о похищении большого объема информации существенно влияют на репутацию организации, косвенно приводя к потерям в объемах торговых операций;
- если похищение осталось незамеченным, конкуренты могут воспользоваться этим с целью полного разорения организации, навязывая ей фиктивные или убыточные соглашения;
- к убыткам может привести и простая подмена информации как на этапе ее передачи, так и на этапе хранения внутри организации.

Совершенно очевидно, что атаки на информацию могут причинить и огромный моральный вред, учитывая возможную конфиденциальность сведений.

По данным исследовательской группы CNews Analytics, специализирующейся на исследованиях рынков информационных технологий и телекоммуникаций, глобальный экономический убыток от компьютерных преступлений, которые так или иначе связаны с атакой на информацию, в 2002 г. составил около 49,2 млрд.$, а в 2003 г. только за август— $8,23 млрд. (16160 атак). В 2004 г., считают аналитики этой же группы, суммарные экономические потери составили от $300 млрд. до $400 млрд. [38]. Учитывая общую тенденцию можно сделать вывод о возрастании количества компьютерных преступлений и в нашей стране.

Исследовательский центр DataPro Research Corp. [39], приводит следующую картину распределения основных причин потери или повреждения информации в компьютерных сетях:

- 52% — неумышленные действия персонала;
- 10% — отказ оборудования;
- 15%) — пожары;
- 10% — затопления.

Что же касается преднамеренных действий, то, по данным того же исследования, главный мотив компьютерных преступлений — похищение денег с электронных счетов (44%). Далее следуют похищение секретной информации (16%), повреждение программного обеспечения (16%), фальсификация информации (12%), заказ услуг на чужой счет (10%). Среди тех, кто был исполнителем указанных действий,— текущий кадровый состав учреждений (81%), посторонние лица (13%), бывшие работники этих же учреждений (6%).

Ясно, что в такой ситуации особое внимание должно отводиться созданию информационных систем, защищенных от разнообразных угроз. Но при этом в процессе проектирования и создания таких систем возникает целый ряд проблем, основными из которых являются:

- сложность интеграции определенных функций безопасности в элементы архитектуры системы;
- сложность формирования исчерпывающего набора необходимых и достаточных требований безопасности;
- отсутствие общепринятых методов проектирования систем безопасности.

Особенностями проектирования архитектуры системы обеспечения безопасности информации, как, наверное, и других сложных многофакторных систем, является множественность параметров, которые необходимо учитывать, и которые тяжело зафиксировать; постоянный рост количества угроз информационной безопасности. При этом большинство существующих решений (например, таких как экранирование и VPN) учитывают только часть проблем обеспечения безопасности информации, а используемые методы, в основном, сводятся к использованию определенного перечня продуктов.

## 1.2. Категории информационной безопасности

Информация, с точки зрения информационной безопасности, характеризуется следующими категориями:

- *конфиденциальность* — гарантия того, что конкретная информация является доступной только тому кругу лиц, для которого она предназначена; нарушение этой категории является *похищением* или *раскрытием информации;*
- *целостность* — гарантия того, что на данный момент информация существует в изначальном виде, то есть, при ее хранении или передаче не было сделано несанкционированных изменений; нарушение данной категории называется *фальсификацией сообщения:*
- *аутентичность* — гарантия того, что источником информации является именно тот субъект, который заявлен как ее автор; нарушение этой категории называется *фальсификацией автора сообщения;*
- *апеллированность* — гарантия того, что в случае необходимости можно доказать, что автором сообщения является именно заявленный субъект и никто другой; отличие данной категории от предыдущей в том, что при подмене автора, кто-то другой пытается заявить об авторстве сообщения, а при нарушении апеллированности сам автор пытается избежать ответственности за выданное им сообщение.

Относительно информационных систем используются другие категории:

- *надежность* — гарантия того, что система будет вести себя запланированно как в нормальном, так и во внештатном режимах;
- *точность* — гарантия точного и полного исполнения всех команд;
- *контроль доступа* — гарантия того, что разные группы лиц имеют разный доступ к информационным объектам, и эти ограничения доступа постоянно исполняются;
- *контролируемость* — гарантия того, что в любой момент может быть выполнена полноценная проверка любого компонента программного комплекса;
- *контроль идентификации* — гарантия того, что клиент (адресат) является именно тем, за кого себя выдает;

- ***устойчивость к умышленным сбоям*** — гарантия того, что при умышленном внесении ошибок в пределах заранее оговоренных норм система будет вести себя прогнозирование.

## 1.3. Возможные варианты защиты информации в автоматизированных системах

Среди возможных методов (мер) защиты конфиденциальной информации наиболее распространенным на сегодня является метод ***криптографической защиты,*** под которым подразумевается скрытие содержания сообщения за счет его ***шифрования (кодирования)*** по определенному алгоритму, цель которого — сделать сообщение непонятным для непосвященных в этот алгоритм или в содержание ключа, использованного при шифровании. Тем не менее, указанный метод защиты не является эффективным по меньшей мере по двум причинам.

Во-первых, зашифрованная с помощью более-менее стойкой криптосистемы информация является недоступной (на протяжении времени, которое определяется стойкостью криптосистемы) для ознакомления без знания алгоритма и ключа. Вследствие этого силовые структуры некоторых стран применяют административные санкции против так называемой "устойчивой криптографии", ограничивая использование криптографических средств частными и юридическими лицами без соответствующей лицензии.

Во-вторых, следует обратить внимание на то, что криптографическая защита защищает лишь содержание конфиденциальной информации. При этом само присутствие шифрованной информации способно привлечь внимание потенциального злоумышленника, который, завладев криптографически защищенным файлом и догадываясь о размещении в нем секретных данных, при определенной мотивации способен направить всю суммарную мощь подконтрольной ему компьютерной сети на дешифрование этих данных. В этом случае проблема информационной безопасности возвращается к устойчивости криптографического кода.

В противовес вышеупомянутому, *стеганографическая* ***защита*** обеспечивает скрытие самого факта существования конфиденциальных сведений при их передаче, хранении или обработке. Под скрытием факта существования подразумевается не только невозможность выявления в перехваченном сообщении наличия другого (скрытого) сообщения, но и вообще невозможность возникновения каких-либо подозрений на этот счет. Задача невозможности неавторизованного извлечения информации при этом отступает на второй план и решается в большинстве случаев дополнительным использованием стандартных криптографических методов. Общей чертой стеганографических методов является то, что скрываемое сообщение встраивается в некий непривлекающий внимания объект (контейнер), который затем открыто транспортируется (пересылается) адресату.

# Глава 2

# Особенности построения стеганографических систем

## 2.1. Предмет, терминология и сферы применения стеганографии

В процессе исследования стеганографии, становится очевидным, что она, по сути, не является чем-то новым. Задачи защиты информации от неавторизированного доступа тем или иным образом решались на протяжении всей истории человечества [2].

За последние десять лет, благодаря массовому распространению мультимедийных технологий и средств телекоммуникаций, развитие стеганографии вышло на принципиально новый этап, который специалисты называют *компьютерной стеганографией* (КС). Среди основных областей использования КС — скрытие (путем встраивания) сообщений в цифровых данных, которые, как правило, имеют аналоговую природу (речь, изображение, аудио- или видеозаписи). В качестве контейнеров (или так называемых "носителей") возможно также использование текстовых файлов или исполняемых файлов программ [I, 3-5, 19, 20]. Так, например, наименее значимые биты цифрового изображения или аудиофайла могут быть заменены данными из текстового файла таким образом, что посторонний независимый наблюдатель не обнаружит никакой потери в качестве изображения или звука [3, 9, 14, 21-23, 27].

Таким образом, скажем, изображение размещенное в определенном Internet-ресурсе общего пользования, потенциально может тайно содержать важную для определенных кругов информацию и при этом не вызывать никаких подозрений широких масс. Публикации некоторых СМИ после сентябрьских терактов в США в 2001 г. даже указывали на данную технику сокрытия как возможный способ связи между членами террористических организаций, которые планировали атаки в знак протеста против влияния Запада на мировой порядок [28-30].

Несмотря на многочисленные открытые публикации и ежегодные конференции, длительное время стеганография не имела сложившейся терминологии. С середины 80-х гг. прошлого столетия для описания модели стеганографической системы (сокращенно — *стегосжтемы* или, что по мнению авторов этой работы является более правильным определением, *стеганосистемы,* поскольку приставка "стего" в переводе с латыни означает "крыша" или "черепица" и искажает смысл используемого понятия) использовалась так называемая "проблема заключенных", которую предложил в 1983 г. Симмонс (G.J. Simmons) [7].

Основные понятия стеганографии были согласованы в 1996 г. на 1-й Международной конференции по скрытию данных — information Workshop on information Hiding*96 [10]. Тем не менее даже такое основополагающее понятие как "стеганография" разными специалистами трактуется неодинаково. Например, некоторые специалисты понимают под стеганографией только скрытую передачу информации, другие же относят к ней такие приложения как, например, метеорная радиосвязь,

радиосвязь с псевдослучайным перестраиванием частоты, широкополосную радиосвязь [31, 32].

В работе [5] приводится следующее определение *цифровой стеганографии:* "...наука о незаметном и надежном скрытии одних битовых последовательностей в других, имеющих аналоговую природу". Упоминанием об аналоговой природе цифровых данных подчеркивается факт встраивания информации в оцифрованные непрерывные сигналы. Таким образом, в сравнении с ЦС, *компьютерная стеганография* имеет более широкий смысл, поскольку в ее пределах рассматриваются вопросы ввода данных в заголовки IP-пакетов, в текстовые сообщения и файлы других форматов.

Слово "незаметное" в представленном выше определении цифровой стеганографии подразумевает обязательное включение человека в систему стеганографической передачи данных. То есть, человек рассматривается как специфический приемник данных, предъявляющий к системе передачи требования, которые достаточно тяжело формализовать [5].

Таким образом, стеганографическая система или, сокращенно, *стеганосистема* — это совокупность средств и методов, которые используются с целью формирования скрытого (незаметного) канала передачи информации. Причем процесс скрытия данных, подобно процессу компрессии (уплотнения), отличается от операции шифрования. Его целью является не ограничивать или регламентировать доступ к сигналу (файлу)-контейнеру, а в значительной степени гарантировать, что встроенные данные останутся неповрежденными (немодифицированными) и подлежащими восстановлению.

При построении стеганосистемы должны учитываться следующие положения [3,5,11]:

- стеганосистема должна иметь приемлемую вычислительную сложность реализации (под вычислительной сложностью понимается количество шагов или арифметико-логических операций, необходимых для решения вычислительной проблемы, в данном случае — процесса встраивания/извлечения конфиденциальной информации в/из сигнала контейнера;
- должна обеспечиваться необходимая пропускная способность (что особенно актуально для стеганосистем скрытой передачи данных);
- методы скрытия должны обеспечивать аутентичность и целостность секретной информации для авторизованного лица;
- потенциальный нарушитель имеет полное представление о стеганосистеме и детали ее реализации; единственное, что ему неизвестно, — это ключ, с помощью которого только его обладатель может установить факт наличия и содержание скрытого сообщения;
- если факт существования скрытого сообщения становится известным нарушителю, это не должно позволить последнему извлечь его до тех пор, пока ключ сохраняется в тайне;
- нарушитель должен быть лишен любых технических и других преимуществ в распознании или, по крайней мере, раскрытии содержания секретных сообщений.

С точки зрения авторов [5], стеганография включает в себя следующие направления:

- встраивание информации с целью ее скрытой передачи;
- встраивание цифровых водяных знаков (ЦВЗ);

- встраивание идентификационных номеров;
- встраивание заголовков.

Анализ других литературных источников, в частности [3, 14, 16, 19-21], и ресурсов Internet позволяет сделать вывод, что на сегодняшний день стеганосистемы активно используются для решения следующих ключевых задач:

- защита конфиденциальной информации от несанкционированного доступа;
- защита авторского права на интеллектуальную собственность;
- преодоление систем мониторинга и управления сетевыми ресурсами;
- "камуфлирование" программного обеспечения;
- создание скрытых от законного пользователя каналов утечки информации.

Сфера **защиты конфиденциальной информации от несанкционированного доступа** является наиболее эффективной при решении проблемы защиты секретной информации. Например, только одна секунда оцифрованного стереозвука с частотой дискретизации 44,1 кГц и уровнем квантования 8 бит за счет замены наименее значимых младших разрядов чисел, характеризующих отсчеты уровней звукового сигнала, битами скрытого сообщения позволяет скрыть около 11 Кбайт информации (при объеме аудиофайла ~ 88,2 Кбайт). При этом изменение результирующего уровня аудиосигнала, соответствующего модифицированному отсчету, составляет менее 0,8%, что при прослушивании не определяется подавляющим большинством людей. Если же звук 16-битный, то изменение уровней вообще составит менее 0.005%.

Другой важной задачей стеганографии является **защита авторского права** от так называемого "'пиратства"'. На компьютерные графические изображения наносится специальная метка, которая остается невидимой для человека, но распознается специализированным программным обеспечением (ПО). Данное направление предназначено не только для обработки изображений, но и для файлов с аудио- или видеоинформацией, и должно обеспечить защиту интеллектуальной собственности.

Стеганографические методы, направленные на противодействие **системам мониторинга и управления сетевыми ресурсами** промышленного шпионажа, позволяют противодействовать способам контроля информационного пространства при прохождении информации через серверы, управляющие локальными и глобальными вычислительными сетями.

Еще одной областью использования стеганосистем является **камуфлирование ПО. В** тех случаях, когда использование ПО незарегистрированными пользователями нежелательно, оно может быть закамуфлировано под стандартные универсальные программные продукты (например, текстовые редакторы) или спрятано в файлах мультимедиа (например, в звуковом сопровождении видео).

Как бы ни отличались направления использования стеганографии, выдвигаемые при этом требования во многом остаются неизменными (см. выше). Однако существуют и отклонения от общепринятых правил. Так, например, отличие постановки задачи для скрытой передачи данных от постановки задачи встраивания ЦВЗ состоит в том, что в первом случае нарушитель должен обнаружить скрытое сообщение, тогда как во втором случае его существование не скрывается. Более того, нарушитель на законных основаниях может иметь устройство для определения ЦВЗ (например, в составе DVD-проигрывателя).

Потенциально возможные сферы использования стеганографии изображены на рис. 2.1.


**Рис. 2.1. Потенциальные области использования стеганографии**

## 2.2. Проблема устойчивости стеганографических систем

Каждая из указанных выше задач требует определенного соотношения между устойчивостью встроенного сообщения к внешним влияниям и размером встроенного сообщения.

Для большинства современных методов, которые используются для скрытия сообщений в файлах цифрового формата, имеет место зависимость надежности системы от объема встраиваемых данных, представленная на рис. 2.2.


**Рис. 2.2. Взаимосвязь между устойчивостью стеганосистемы и объемом скрываемого сообщения при неизменном размере файла-контейнера**

Из рис. 2.2 видно, что увеличение объема встраиваемых данных значительно снижает надежность системы.

Таким образом, существует перспектива принятия оптимального решения при выборе между количеством (объемом) скрываемых данных и степенью устойчивости (скрытости) к возможной модификации (анализу) сигнала-контейнера. Путем ограничения степени ухудшения качеств контейнера, которые способен воспринимать человек, при стеганографической обработке контейнера можно достичь или высокого уровня (объема) встраиваемых данных, или высокой устойчивости к модификации (анализу), но никоим образом не обоих этих показателей одновременно, поскольку рост одного из них неизбежно приводит к уменьшению другого. Несмотря на то, что данное утверждение математически может быть продемонстрировано только для некоторых методов стеганографии (например, для скрытия путем рас-

ширения спектра), очевидно, что оно является справедливым и для других методов скрытия данных [14].

При использовании любого метода, благодаря избыточности информации, существует возможность повысить степень надежности скрытия, жертвуя при этом пропускной способностью (объемом скрываемых данных). Объем встроенных данных и степень модификации контейнера изменяется от метода к методу. Также очевиден и тот факт, что в зависимости от целей, для которых используется скрытие данных, различными являются и требования относительно уровня устойчивости системы к модификации контейнера. Как следствие, для разных целей оптимальными являются разные методы стеганографии.

Рассмотрим процесс проведения *стеганоанализа* — оценки перехваченного контейнера на предмет наличия в нем скрытого сообщения. Скрытие информации внутри электронного носителя требует изменений (перестройки) свойств последнего, что в той или иной степени приводит к ухудшению его характеристик или к обретению этими характеристиками несвойственных им значений. Данные характеристики могут выполнить роль "подписей"', которые сигнализируют о существовании встроенного сообщения, и, таким образом, основная идея стеганографии — скрытие факта существования секретной информации — не будет выполненной.

Стеганоанализ на предмет наличия скрытой информации может приобретать разные формы: обнаружение наличия (детектирование), извлечение и, наконец, удаление или разрушение скрытых данных [40]. Кроме того, нарушитель может поверх уже существующей скрытой информации встроить определенную дезинформацию. Более детально атаки на стеганосистемы и противодействие им рассмотрены в главе 3.

## 2.3. Структурная схема и математическая модель типичной стеганосистемы

В общем случае стеганосистема может быть рассмотрена как система связи [5] Обобщенная структурная схема стеганосистемы изображена на рис. 2.3.

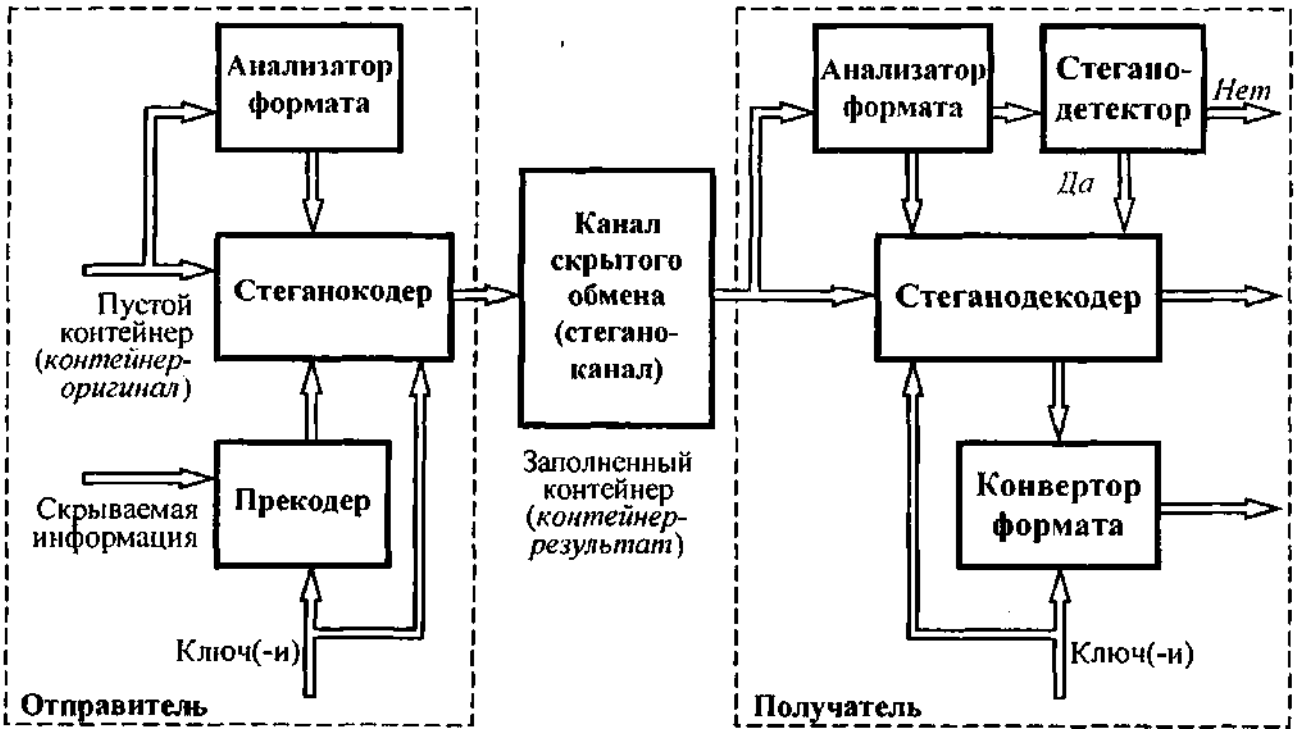



Рис. 2.3. Структурная схема стеганосистемы как системы связи

Основными стеганографическими понятиями являются *сообщение* и *контейнер*. *Сообщение* $m \in M$ это секретная информация, наличие которой необходимо скрыть, $M = \{m_1, m_2, ..., m_n\}$ — множество всех сообщений.

*Контейнером с* е С называется несекретная информация, которую можно использовать для скрытия сообщения, $C = \{c_1, c_2, ..., c_q\}$ — множество всех контейнеров, причем $q >> n$. В качестве сообщения и контейнера могут выступать как обычный текст, так и файлы мультимедийного формата.

*Пустой контейнер* (или так называемый *контейнер-оригинал)* — это контейнер с, который не содержит скрытой информации. *Заполненный контейнер (контейнер-результат)*— контейнер с\ который содержит скрытую информацию *т (с,„)*. Одно из требований, которое при это ставится: контейнер-результат не должен быть визуально отличим от контейнера-оригинала. Выделяют два основных типа контейнера: *потоковый* и *фиксированный*.

*Потоковый контейнер* представляет собой последовательность битов, которая непрерывно изменяется. Сообщение встраивается в него в реальном масштабе времени, поэтому в кодере заранее неизвестно, хватит ли размеров контейнера для передачи всего сообщения. В один контейнер большого размера может быть встроено несколько сообщений. Интервалы между встроенными битами определяются генератором ПСП с равномерным распределением интервалов между отсчетами.

Основная проблема заключается в выполнении синхронизации, определении начала и конца последовательности. Если в данных контейнера существуют биты синхронизации, заголовки пакетов и т.д., то скрытая информация может следовать сразу же после них. Сложность организации синхронизации является преимуществом с точки зрения обеспечения скрытости передачи. К сожалению, на сегодняшний день практически отсутствуют работы, посвященные разработке стеганосистем с потоковым контейнером.

В качестве примера перспективной реализации потокового контейнера можно привести стеганоприставку к обычному телефону. При этом под прикрытием заурядного, несущественного телефонного разговора можно передавать другой разговор, данные и т.д. Не зная секретного ключа, нельзя не только узнать о содержании скрытой передачи, но и о самом факте ее существования.

В *фиксированном контейнере* размеры и характеристики последнего заранее известны. Это позволяет выполнять вложение данных оптимальным (в определенном смысле) образом. Далее будут рассматриваться преимущественно фиксированные контейнеры (в дальнейшем — просто "контейнеры").

Контейнер может быть избранным, случайным или навязанным. Избранный контейнер зависит от встроенного сообщения, а в предельном случае является его функцией. Такой тип контейнера больше характерен именно *для* стеганографии. Навязанный контейнер появляется, когда лицо, которое предоставляет контейнер, подозревает о возможной скрытой переписке и желает предотвратить ее. На практике же чаще всего имеют дело со случайным контейнером [5].

Скрытие информации, которая преимущественно имеет большой объем,, выдвигает существенные требования к контейнеру, размер которого должен по меньшей мере в несколько раз превышать размер встраиваемых данных. Понятно, что для увеличения скрытости указанное соотношение должно быть как можно большим.

Перед тем как выполнить вложение сообщения в контейнер, его необходимо преобразовать в определенный удобный для упаковки вид. Кроме того, перед упаковкой в контейнер, для повышения защищенности секретной информации последнюю можно зашифровать достаточно устойчивым криптографическим кодом [14].

Во многих случаях также желательна устойчивость полученного стеганосообщения к искажениям (в том числе и злоумышленным) [5].

В процессе передачи звук, изображение или какая-либо другая информация, используемая в качестве контейнера, может подвергаться разным трансформациям (в том числе с использованием алгоритмов с потерей данных): изменение объема, преобразование в другой формат и т.п.. — поэтому для сохранения целостности встроенного сообщения может понадобиться использование кода с исправлением ошибок (помехоустойчивое кодирование).

Начальную обработку скрываемой информации выполняет изображенный на рис. 2.3 *прекодер*. В качестве одной из важнейших предварительных обработок сообщения (а также и контейнера) можно назвать вычисление его обобщенного преобразования Фурье. Это позволяет осуществить встраивание данных в спектральной области, что значительно повышает их устойчивость к искажениям.

Следует отметить, что для увеличения секретности встраивания, предварительная обработка довольно часто выполняется с использованием ключа.

Упаковка сообщения в контейнер (с учетом формата данных, представляющих контейнер) выполняется с помощью *стеганокодера*. Вложение происходит, например, путем модификации наименьших значащих битов контейнера. Вообще, именно алгоритм (стратегия) внесения элементов сообщения в контейнер определяет *методы* стеганографии, которые в свою очередь делятся на определенные *группы,* например, в зависимости от того, файл какого формата был выбран в качестве контейнера.

В большинстве стеганосистем для упаковки и извлечения сообщений используется *ключ,* который предопределяет секретный алгоритм, определяющий порядок внесения сообщения в контейнер. По аналогии с криптографией, тип ключа обуславливает существование двух типов стеганосистем:

- *с секретным ключом* — используется один ключ, который определяется до начала обмена стеганограммой или передается защищенным каналом;
- *с открытым ключом* — для упаковки и распаковки сообщения используются разные ключи, которые отличаются таким образом, что с помощью вычислений невозможно получить один ключ из другого, поэтому один из ключей (открытый) может свободно передаваться по незащищенному каналу.

В качестве секретного алгоритма может быть использован *генератор псевдослучайной последовательности* (ПСП) битов. Качественный генератор ПСП, ориентированный на использование в системах защиты информации, должен соответствовать определенным требованиям [42]. Перечислим некоторые из них.

- Криптографическая стойкость — отсутствие у нарушителя возможности предусмотреть следующий бит на основании известных ему предыдущих с вероятностью, отличной от 1/2. На практике криптографическая стойкость оценивается статистическими методами. Национальным Институтом Стандартов и Технологий США (НИСТ) разработано Руководство по проведению статистических испытаний генераторов ПСП, ориентированных на использование в задачах криптографической защиты информации [43]).
- Хорошие статистические свойства — ПСП по своим статистическим свойствам не должна существенно отличаться от истинно случайной последовательности.
- Большой период формированной последовательности.
- Эффективная аппаратно-программная реализация.

Статистически (криптографически) безопасный генератор ПСП должен соответствовать следующим требованиям:

- ни один статистический тест не определяет в ПСП никаких закономерностей, иными словами, не отличает эту последовательность от истинно случайной;
- при инициализации случайными значениями генератор порождает статистически независимые псевдослучайные последовательности.

В качестве основы генератора может использоваться, например, линейный рекуррентный регистр. Тогда адресатам для обеспечения связи должно сообщаться начальное заполнение этого регистра. Числа, порождаемые генератором ПСП, могут определять позиции модифицированных отсчетов в случае фиксированного контейнера или интервалы между ними в случае потокового контейнера.

Следует отметить, что метод случайного выбора величины интервала между встроенными битами не является достаточно эффективным по двум причинам. Во-первых, скрытые данные должны быть распределены по всему контейнеру, поэтому равномерное распределение длины интервалов (от наименьшего к наибольшему) может быть достигнуто только приблизительно, поскольку должна существовать уверенность в том, что все сообщение встроено (то есть, поместилось в контейнер). Во-вторых, длина интервалов между отсчетами шума (во многих моделях сигнал-контейнер рассматривается как аддитивный шум [44]) распределена не по равномерному, а по экспоненциальному закону. Генератор ПСП с экспоненциальным распределением интервалов сложен в реализации.

Скрываемая информация заносится в соответствии с ключом в те биты, модификация которых не приводит к существенным искажениям контейнера. Эти биты образуют так называемый *стеганопуть*. Под "существенным" подразумевается искажение, которое приводит к росту вероятности выявления факта наличия скрытого сообщения после проведения стеганоанализа.

*Стеганографический канал* — канал передачи контейнера-результата (вообще, существование канала как, собственно говоря, и получателя — наиболее обобщенный случай, поскольку заполненный контейнер может, например, храниться у "отправителя", который поставил перед собой цель ограничить неавторизованный доступ к определенной информации. В данном случае отправитель выступает в роли получателя). Во время пребывания в стеганографическом канале контейнер, содержащий скрытое сообщении, может подвергаться умышленным атакам или случайным помехам, детальное описание которых представлено в главе 3.

В *стеганодетекторе* определяется наличие в контейнере (возможно уже измененном) скрытых данных. Это изменение может быть обусловлено влиянием ошибок в канале связи, операций обработки сигнала, намеренных атак нарушителей. Как уже отмечалось выше, во многих моделях стеганосистем сигнал-контейнер рассматривается как аддитивный шум. Тогда задача выявления и выделения стеганосообщения является классической для теории связи [70]. Но такой подход не учитывает двух факторов: неслучайного характера контейнера и требований по сохранению его качеств. Эти моменты не встречаются в известной теории обнаружения и выделения сигналов на фоне аддитивного шума. Очевидно, что их учет позволит построить более эффективные стеганосистемы [5].

Различают стеганодетекторы, предназначенные только для обнаружения факта наличия встроенного сообщения, и устройства, предназначенные для выделения этого сообщения из контейнера, — *стеганодекодеры*.

Итак, в стеганосистеме происходит объединение двух типов информации таким образом, чтобы они по-разному воспринимались принципиально разными детекто-

рами. В качестве одного из детекторов выступает система выделения скрытого сообщения, в качестве другого — человек.

Алгоритм встраивания сообщения в простейшем случае состоит из двух основных этапов:

1. Встраивание в стеганокодере секретного сообщения в контейнер-оригинал.
2. Обнаружение (выделение) в стеганодетекторе (декодере) скрытого зашифрованного сообщения из контейнера-результата.

Исходя из этого, рассмотрим математическую модель стеганосистемы. Процесс тривиального *стеганографического преобразования* описывается зависимостями:

$$E:\ C \times M \rightarrow S; \tag{2.1}$$

$$D:\ S \rightarrow M, \tag{2.2}$$

где $S = \{(c_1, m_1), (c_2, m_2), ..., (c_n, m_n), ..., (c_q, m_q)\} = \{s_1, s_2, ..., s_q\}$ — множество контейнеров-результатов (стеганограмм).

Зависимость (2.1) описывает процесс скрытия информации, зависимость (2.2)— извлечение скрытой информации. Необходимым условием при этом является отсутствие "пересечения" [3], то есть, если $m_a \neq m_b$, причем $m_a, m_b \in M$, а $(c_a, m_a), (c_b, m_b) \in S$, то $E(c_a, m_a) \cap E(c_b, m_b) = \varnothing$. Кроме того, необходимо, чтобы мощность множества $|C| \geq |M|$. При этом оба адресата (отправитель и получатель) должны знать алгоритм прямого *(E)* и обратного *(D)* стеганографического преобразования.

Итак, в общем случае *стеганосистема* — это совокупность $\Sigma$ = (C, *M, S, E, D)* контейнеров (оригиналов и результатов), сообщений и преобразований, которые их связывают.

Для большинства стеганосистем множество контейнеров C выбирается таким образом, чтобы в результате стеганографического преобразования (2.1) заполненный контейнер и контейнер-оригинал были подобны, что формально может быть оценено с помощью функции подобия [3].

| Определение 2.1

Пусть *C* — непустое множество, тогда функция $sim(C) \rightarrow (-\infty, 1]$ является *функцией подобия* на множестве C, если для каких-либо *x,y e C* справедливо, что $sim(x, y) = 1$ в случае $x = y$ и $sim(x, y) < 1$ при $x \neq y$

Стеганосистема может считаться *надежной,* если $sim[c, E(c, m)] \approx 1$ для всех $m \in M$ i $c \in C$, причем в качестве контейнера *c* должен избираться такой, который ранее не использовался. Кроме того, неавторизованное лицо не должно иметь доступ к набору контейнеров, используемых для секретной связи.

Выбор определенного контейнера *c* из набора возможных контейнеров *C* может осуществляться произвольно (так называемый *суррогатный метод* выбора контейнера) или путем избрания наиболее пригодного, который менее других изменится во время стеганопреобразования *(селективный метод).* В последнем случае контейнер избирается в соответствии с правилом:

$$c = \max_{x \in C} sim[x, E(x, m)] \tag{2.3}$$

Также следует отметить, что функции прямого *(E)* и обратного (D) стеганографического преобразования в общем случае могут быть произвольными (но, конечно, соответствующими одна другой), однако на практике требования к устойчиво-

сти скрытой информации накладывают на указанные функции определенные ограничения.

Так, в подавляющем большинстве случаев, $E(c, m) \approx E(c + \delta, m)$I, или $D[E(c, m)] \approx D[E(c + \delta, m)] = m$, то есть, незначительно модифицированный контейнер (на величину δ) не должен приводить к изменению скрытой в нем информации [5].

## 2.4. Протоколы стеганографических систем

Важное значение для достижения целей стеганографии имеют протоколы. Под *протоколом* подразумевается "порядок действий, к которым прибегают две или несколько сторон, предназначенный для решения определенной задачи'* [11]. Можно разработать исключительно эффективный алгоритм скрытия информации, но из-за неправильного его применения не достичь своей цели.

И протокол, и алгоритм являются определенной последовательностью действий. Отличие между ними заключается в том, что к протоколу должны быть обязательно привлечены две или более сторон. При этом допускается, что участники принимают на себя обязательства придерживаться протокола. Так же как и алгоритм, протокол состоит из шагов. На каждом шаге протокола выполняются определенные действия, которые могут заключаться, например, в проведении некоторых вычислений.

Как уже отмечалось в предыдущем подразделе, в стеганографии различают системы с секретным ключом и системы с открытым ключом. В первых используется один ключ, который должен быть заранее известен авторизованным абонентам до начала скрытого обмена секретными сообщениями (или же переслан защищенным каналом во время указанного обмена). В системах с открытым ключом для встраивания и извлечения скрытой информации используются разные, не выводимые один из другого ключи — открытый и секретный.

Учитывая большое разнообразие стеганографических систем, целесообразно свести их к следующим четырем типам [3]:

- бесключевые стеганоеистемы;
- системы с секретным ключом;
- системы с открытым ключом;
- смешанные стеганосистемы.

### *2.4.1. Бесключевые стеганосистемы*

*Для* функционирования бесключевых стеган ос истем, кроме алгоритма стеганографического преобразования, отсутствует необходимость в дополнительных данных, наподобие стеганоключа.

Определение 2.2

Совокупность $\Sigma = (C, M, S, E, D)$, где $C$ — множество контейнеров-оригиналов; $M$ — множество секретных сообщений, причем $|M| \le |C|$; $S$ — множество контейнеров-результатов, причем $sim(C, S) \to 1$; $E: C \times M \to S$ и $D: S \to M$ — соответственно функции прямого (встраивание) и обратного (извлечение) стеганопреобразований, причем $D[E(c, m)] = m$ для любых $m \in M$ и $c \in C$, называется *бесключевой стеганографической системой.*

Таким образом, безопасность бесключевых стеганосистем базируется только на секретности используемых стеганографических преобразований *E* и *D*. Это противоречит определяющему принципу, который установил Керхгофс (A. Kerckhoffs) для систем защиты информации [6], поскольку стойкость системы зависит только от степени проинформированности нарушителя относительно функций $E$ и $D$.

Для повышения безопасности бесключевых систем перед началом процесса стеганографического скрытия предварительно выцолняется криптографическое шифрование скрываемой информации. Совершенно очевидно, что такой подход увеличивает защищенность всего процесса связи, поскольку усложняет выявление скрытого сообщения. Однако "сильные" стеганосистемы, как правило, способны выполнять возложенные на них функции без предварительной криптографической защиты встроенного сообщения.

## *2,4.2, Стеганосистемы с секретным ключом*

По принципу Керхгофса, безопасность системы должна базироваться на определенном фрагменте секретной информации — ключе, который (как правило, предварительно) разделяется между авторизованными лицами. Отправитель, встраивая секретное сообщение в избранный контейнер с, использует стеганоключ *к*. Если получатель знает данный ключ, то он может извлечь из контейнера скрытое сообщение. Без знания ключа любое постороннее лицо этого сделать не сможет.

**Определение 2.3**

***Стеганосистемой*** *с* ***секретным*** *ключом* называют совокупность $\Sigma = (C, M, K, S^K, E, D)$, где $C$ — множество контейнеров-оригиналов; $M$— множество секретных сообщений, причем $|M| \le |C|$; $S^K$ — множество контейнеров-результатов, причем $sim(C, S^K) \to 1$; $K$— множество секретных стеганоключей; $E: C \times M \times K \to S^K$ и $D: S^K \times K \to M$ — функции прямого и обратного стеганопреобразования со свойством $D[E(c, m, k), k] = m$ для любых $m \in M$, $c \in C$ и $k \in K$.

Данный тип стеганосистем предполагает наличие безопасного (защищенного) канала обмена стеганоключами.

Иногда ключ $k$ вычисляют с помощью секретной хэш-функции (hash function), используя некоторые характерные черты контейнера. Если стеганопреобразование *E* не изменяет в окончательной стеганограмме выбранные особенности контейнера, то получатель также сможет вычислить стеганоключ (хотя и в этом случае защита будет зависеть от секретности хэш-функции, и, таким образом, опять нарушается принцип Керхгофса). Очевидно, что для достижения адекватного уровня защиты такую особенность в контейнере необходимо выбирать достаточно внимательно.

В некоторых алгоритмах во время извлечения скрытой информации дополнительно необходимы сведения о первичном (пустом) контейнере или некоторые другие данные, отсутствующие в стеганограмме. Такие системы представляют ограниченный интерес, поскольку они требуют передачи изначального вида контейнера, что эквивалентно традиционной задаче обмена ключами. Подобные алгоритмы могут быть отмечены как отдельный случай стеганосистем с секретным ключом, у которых $K = C$ или $K = C \times K'$, где $K'$ — множество дополнительных наборов секретных ключей.

## 2.4.3. Стеганосистемы с открытым ключом

Стеганография с открытым ключом опирается на достижения криптографии последних 30 лет. Стеганографические системы с открытым ключом не имеют потребности в дополнительном канале ключевого обмена. Для их функционирования необходимо иметь два стеганоключа: один секретный, который необходимо хранить в тайне, а другой — открытый, который может храниться в доступном для всех месте. При этом открытый ключ используется для встраивания сообщения, а секретный — для его извлечения.

**Г Определение 2.4 |**

***Стеганосистемой с открытым ключом*** называют совокупность $\Sigma = (C, M, K, S^K, E, D)$, где $C$ — множество контейнеров-оригиналов; $M$ — множество секретных сообщений, $|M| \le |C|$; $S^K$ — множество контейнеров-результатов; $sim(C, S^K) \to 1$; $K = (k_O, k_C)$ — множество пар стеганоключей (открытый ключ $k_O$ используется для скрытия информации, а секретный ключ $k_C$ для ее извлечения); $E: C \times M \times k_O \to S^K$ и $D: S^K \times k_C \to M$ — функции прямого и обратного стеганопреобразования со свойством $D[E(c, m, k_O), k_C] = m$ для любых $m \in M$, $c \in C$ и $k_O, k_C \in K$.

Следует отметить, что стеганоключ не шифрует данные, а скрывает место их встраивания в контейнере. Скрытые данные могут быть дополнительно зашифрованы классическим методом, но этот вопрос не касается непосредственно стеганографии.

Стеганосистемы с открытым ключом используют тот факт, что функция извлечения скрытых данных $D$ может быть применена к любому контейнеру, независимо от того, находится в нем скрытое сообщение или нет *{c-,* или *si).* Если скрытое сообщение отсутствует, то на выходе будет получена некоторая случайная последовательность. Если эта последовательность статистически не отличается от шифртекста криптосистемы с открытым ключом, тогда в безопасной стеганоеистеме можно скрывать полученный таким образом шифртекст, а не открытый текст [3].

## 2.4.4. Смешанные стеганосистемы

На практике преимущество отдается бесключевым стеганосистемам, хотя последние могут быть раскрыты в случае, если нарушитель узнает о методе стеганопреобразования, который был при этом использован. В связи с этим в бесключевых системах часто используют особенности криптографических систем с открытым и/или секретным ключом [3,12].

Учитывая большое разнообразие форматов, которые могут иметь скрываемые сообщения и контейнеры (текст, звук или видео, которые, в свою очередь, также делятся на подформаты), представляется целесообразным предварительное преобразование сообщения в удобный для встраивания и оптимальный с точки зрения скрытости в заданном контейнере формат [5]: $U: C \times M \times K \to W$, $w = U(c, m, k)$. Другими словами, необходимо учитывать как особенности встраиваемого сообщения, так и особенности контейнера, в который планируется его ввести.

Произвольность функции $U$ ограничивается требованиями устойчивости к разного рода влияниям на полученный контейнер-результат. Кроме того, функция $U$ является составной:

$$U = T \circ G, \tag{2.4}$$

где $G: M \times K \to Z$; $T: C \times Z \to W$.

Функция $G$ может быть реализована, например, с помощью криптографического безопасного генератора ПСП с $K$ в качестве изначального значения. Для повышения устойчивости скрытого сообщения могут использоваться помехоустойчивые коды, например, коды Хемминга, БЧХ, Голея, сверточные коды [65].

Оператор $T$ модифицирует кодовые слова $Z$ с учетом формата контейнера, в результате чего получается оптимальное для встраивания сообщение. Функция $T$ должна быть выбрана таким образом, чтобы контейнер-оригинал C, контейнер-результат $S$ и модифицированный в предусмотренных границах контейнер-результат $\tilde{S}$ порождали одно и то же оптимальное для встраивания сообщение:

$$T: C \times Z = T: S \times Z = T: \tilde{S} \times Z = \rightarrow W. \quad (2.5)$$

Процесс встраивания сообщения $W$ в контейнер-оригинал $C$ при этом можно описать как суперпозицию сигналов:

$$E: C \times V \times W \rightarrow S; s(x, y) = c(x, y) * v(x, y)\, w(x, y)\, p(x, y), \quad (2.6)$$

где $\nu(\chi,y)$ — маска встраивания сообщений, которая учитывает, например, характеристики зрительной системы среднестатистического человека и служит для уменьшения заметности этих сообщений; $p(x,y)$ — проецирующая функция, которая зависит от ключа; знак "*" означает оператор суперпозиции, который в общем случае включает в себя, кроме сложения, ограничение уровня и квантование.

Проецирующая функция осуществляет "распределение" оптимизированного сообщения по всей области контейнера. Ее использование может рассматриваться как реализация разнесения конфиденциальной информации параллельными каналами. Кроме того, эта функция имеет определенную пространственную структуру и корреляционные свойства, что используется для противодействия, например, геометрическим атакам (см. главу 3).

Еще одно возможное описание процесса встраивания представлено в [5] со ссылкой на [41]. Представим стеганографическую систему как систему связи с передачей дополнительной информации (рис. 2.4).

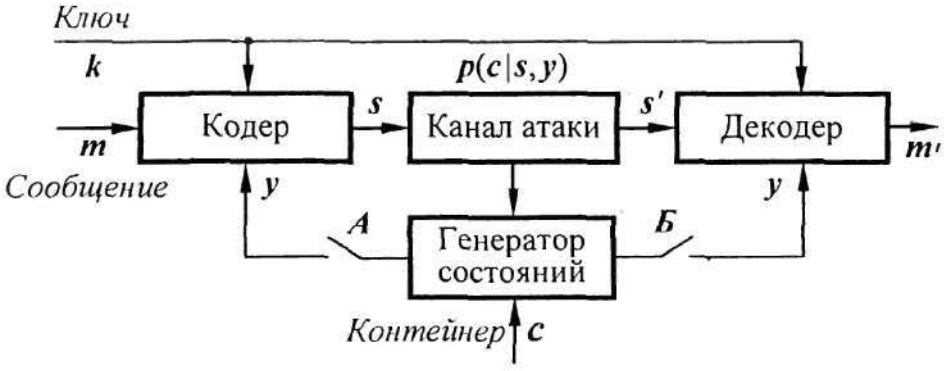



Рис. 2.4. Представление стеганосистемы как системы связи с передачей дополнительной информации

В этой модели кодер и декодер имеют доступ, кроме ключа, еще и к информации о канале (то есть о контейнере и о возможных атаках). В зависимости от положения переключателей $A$ и $Б$ выделяют четыре класса стеганосистем (при этом считается, что ключ всегда известен кодеру и декодеру).

- I класс. Дополнительная информация отсутствует (переключатели разомкнуты) — так называемые "классические" стеганосистемы. В ранних работах по стеганографии считалось, что информация о канале является недоступной коде-

ру. Выявление скрытой информации осуществлялось путем вычисления коэффициента корреляции между принятым контейнером и вычисленным по ключу сообщением. Если коэффициент превышал некоторый порог, принималось решение относительно присутствия встроенных данных. Но известно, что корреляционный приемник является оптимальным только в случае аддитивной гауссовской помехи [70]. При других атаках (например, геометрических искажениях) данные стеганосистемы давали неудовлетворительные результаты.

- **II класс.** Информация о канале известна только кодеру (ключ *А* замкнут, *Б* разомкнут). Такая конструкция привлекла к себе внимание, благодаря труду [47]. Особенностью схемы является то, что, будучи "слепой", она имеет ту же теоретическую пропускную способность, что и схема с наличием контейнера-оригинала в декодере. К недостаткам стеганосистем класса II можна отнести высокую сложность кодера (необходимость построения кодовой книги для каждого контейнера), а также отсутствие адаптации системы к возможным атакам. В последнее время предложен ряд практических подходов, которые устраняют эти недостатки. В частности, для снижения сложности кодера предлагается использовать структурированные кодовые книги, а декодер рассчитывать на случай наихудшей атаки.
- **III** класс. Дополнительная информация известна только декодеру (переключатель *А* разомкнут, *Б* замкнут). Декодер строится с учетом возможных атак. В результате получают устойчивые к геометрическим атакам системы. Один из методов достижения этой цели — использование так называемого опорного встроенного сообщения (аналог пилот-сигнала в радиосвязи). Опорное сообщение — небольшое количество бит, встроенных в инвариантные к преобразованиям коэффициенты сигнала. Например, можно выполнить встраивание в амплитудные коэффициенты преобразования Фурье, которые являются инвариантными к афинным (геометрическим) преобразованиям. Тогда опорное сообщение укажет, какое преобразование выполнил над контейнером нарушитель. Другим назначением пилотного сообщения является борьба с замираниями, по аналогии с радиосвязью. Замираниями в данном контексте можно считать изменение значений отсчетов сигнала при встраивании данных, атаках, добавлении негауссовского шума и т.д. В радиосвязи для борьбы с замираниями используется метод разнесенного приема (по частоте, во времени, пространстве, по коду), а в стеганографии используется разнесение встроенных сообщений в пространстве контейнера. Пилотное сообщение генерируется в декодере на основании ключа.
- **IV класс.** Дополнительная информация известна как в кодере, так и в декодере (оба переключателя замкнуты). Как отмечено в [46], все перспективные стеганосистемы должны строиться именно по этому принципу. Оптимальность такой схемы достигается путем оптимального согласования кодера с сигналом-контейнером, а также адаптивным управлением декодером в условиях наблюдения канала атак.

## 2.5. Выводы

В данной главе путем анализа специализированных литературных источников и ресурсов сети Internet представлено обобщенное определение понятия стеганографической системы, определены существующие и перспективные направления, в ко-

торых возможно использование стеганографии как инструмента защиты информации в автоматизированных системах, рассмотрена проблема соотношения между устойчивостью стеганосистем и объемом скрываемого с ее помощью сообщения, раскрыта сущность таких основных понятий стеганографии как сообщение, контейнер-оригинал, контейнер-результат, стеганоключ, стеганоканал и др. Это позволило непосредственно перейти к построению структурной схемы стеганосистемы, что было сделано с позиции теории связи, выполнить систематизированный обзор известных протоколов стеганосистем.

Полученные результаты позволяют сделать вывод о нецелесообразности использования бесключевых стеганосистем, безопасность которых базируется только на секретности используемых стеганографических преобразований. Перечислены преимущества использования открытого ключа в сравнении с секретным ключом. В основной перечень стеганографических протоколов предложено включить протокол, подразумевающий дополнительное предварительное преобразование скрываемой информации к оптимальному формату, исходя из особенностей формата носителя, который планируется использовать в качестве контейнера.

# Глава 3

# Принципы стеганографического анализа

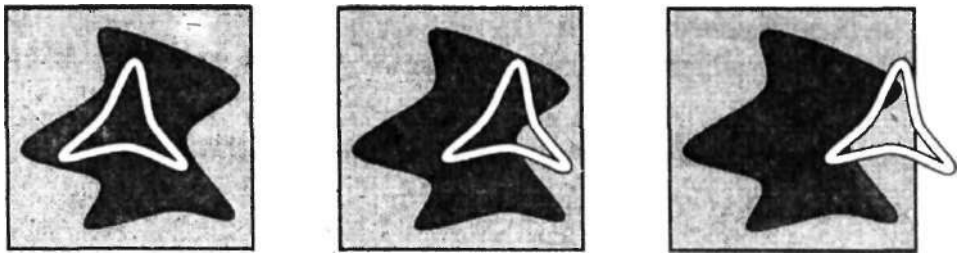
## 3.1. Вступительные положения

Основной целью стеганоанализа является моделирование стеганографических систем и их исследование для получения качественных и количественных оценок надежности использования стеганопреобразования, а также построение методов выявления скрываемой в контейнере информации, ее модификации или разрушения.

Терминология стеганоанализа аналогична терминологии криптоанализа, однако присутствуют и некоторые существенные расхождения. Криптоанализ применяется с целью дешифровки содержания криптограмм, а стеганоанализ — для выявления наличия скрытой информации.

По уровню обеспечения секретности, стеганосистемы делятся на теоретически устойчивые, практически устойчивые и неустойчивые системы [3].

*Теоретически устойчивая (абсолютно надежная) стеганоеистема* осуществляет скрытие информации только в тех фрагментах контейнера, значение элементов которых не превышает уровень шумов или ошибок квантования, и при этом теоретически доказано, что невозможно создать стеганоаналитический метод выявления скрытой информации (рис. 3.1, а).



*а)* *б)* *в)*

- область *практической* неразличимости, в которой изменения контейнера не распознаются существующими у нарушителя аналитическими методами;
- область *теоретической* неразличимости, в которой скрытые элементы практически не могут быть распознаны, поскольку находятся ниже уровня шумов и ошибок квантования;
- область защиты стеганографической системы.

Рис. 3.1. Соотношение методов стеганозащиты и стеганоанализа: а — теоретически устойчивая стега.юсистема; *б*— практически устойчивая стеганосистема; β — неустойчивая стеганосистема

***Практически устойчивая стеганосистема*** выполняет такую модификацию фрагментов контейнера, изменения которых могут быть выявлены, но при этом известно, что на данный момент необходимые стеганоаналитические методы у нарушителя отсутствуют или пока что не разработаны (рис. 3.1,6).

***Неустойчивая стеганоеистема*** скрывает информацию таким образом, что существующие стеганоаналитические средства позволяют ее выявить (рис. 3.1, в). В этом случае стеганографический анализ помогает найти уязвимые места стеганографического преобразования и усовершенствовать его таким образом, чтобы все изменения, внесенные в контейнер, оказались бы в области теоретической или, по крайней мере, практической неразличимости (рис. 3.1, *а, б).*

Нарушитель может быть пассивным, активным и злонамеренным. В зависимости от этого, он может создавать различные угрозы. *Пассивный нарушитель* способен только обнаружить факт наличия стеганоканала и (возможно) узнать о содержании сообщения. Будет ли он способен "прочитать" сообщение после его обнаружения, зависит от устойчивости системы шифрования, и этот вопрос, как правило, в стеганографии не рассматривается.

Диапазон действия ***активного нарушителя*** значительно шире. Скрытое сообщение может быть им удалено или разрушено. Действия ***злонамеренного*** *нарушителя* наиболее опасны. Он способен не только разрушать, но и создавать фальшивые стеганограммы (дезинформацию) [40]. Для осуществления той или иной угрозы нарушитель применяет атаки.

## 3.2. Виды атак на стеганографическую систему

Стеганосистема считается *взломанной,* если нарушителю удалось, по крайней мере, доказать существование скрытого сообщения в перехваченном контейнере. Предполагается, что нарушитель способен осуществлять любые типы атак и имеет неограниченные вычислительные возможности. Если ему не удается подтвердить гипотезу о том, что в контейнере скрыто секретное сообщение, то стеганографическая система считается ***устойчивой.***

В большинстве случаев выделяют несколько этапов взлома стеганографической системы:

1. Обнаружение факта присутствия скрытой информации.
2. Извлечение скрытого сообщения.
3. Видоизменение (модификация) скрытой информации.
4. Запрет на выполнение любой пересылки информации, в том числе скрытой [40].

Первые два этапа относятся к пассивным атакам на стеганосистему, а последние — к активным (или злонамеренным) атакам. Выделяют следующие виды атак на стеганосистемы (по аналогии с криптоанализом) [3,5]:

- **Атака на основании известного заполненного контейнера.** В этом случае нарушитель имеет в своем распоряжении один или несколько заполненных контейнеров (в последнем случае предполагается, что встраивание скрытой информации выполнялось тем же самым способом). Задача нарушителя может заключаться в выявлении факта наличия стеганоканала (основное задание), а также в извлечении данных или определении ключа. Зная ключ, нарушитель имеет возможность анализа других стеганосообщений.
- **Атака на основании известного встроенного сообщения.** Этот тип атаки в большей мере характерен для систем защиты интеллектуальной собственно-

сти, когда в качестве ЦВЗ, например, используется известный логотип фирмы. Задачей анализа является получение ключа. Если соответствующий скрытому сообщению заполненный контейнер неизвестен, то задача является практически неразрешимой.

- **Атака на основании выбранного скрытого сообщения.** В этом случае нарушитель может предлагать для передачи свои сообщения и анализировать полученные при этом контейнеры-результаты.
- **Адаптивная атака на основании выбранного сообщения.** Эта атака является частным случаем предыдущей. При этом нарушитель имеет возможность выбирать сообщения для навязывания их передачи адаптивно, в зависимости от результатов анализа предшествующих контейнеров-результатов.
- **Атака на основании выбранного заполненного контейнера.** Этот тип атаки более характерен для систем ЦВЗ. У стеганоаналитика есть детектор заполненных контейнеров в виде "черного ящика" и несколько таких контейнеров. Анализируя продетектированные скрытые сообщения, нарушитель пытается раскрыть ключ.

Кроме того, у нарушителя может существовать возможность применять еще три атаки, не имеющих прямых аналогов в криптоанализе:

- **Атака на основании известного пустого контейнера.** Если последний известен нарушителю, то путем сравнения его с подозреваемым на присутствие скрытых данных контейнером он всегда может установить факт наличия стеганоканала. Несмотря на тривиальность этого случая, в ряде изданий приводится его информационно-теоретическое обоснование. Намного более интересным представляется сценарий, когда контейнер известен приблизительно, с некоторой погрешностью (например, если к нему добавлен шум). В этом случае существует возможность построения устойчивой стеганосистемы [5].
- **Атака на основании выбранного пустого контейнера.** В этом случае нарушитель способен заставить воспользоваться предложенным им контейнером. Последний, например, может иметь значительные однородные области (однотонные изображения), и тогда обеспечить секретность встраивания будет непросто.
- **Атака на основании известной математической модели контейнера или его части.** При этом атакующий пытается определить отличие подозреваемого сообщения от известной ему модели. Например, можно допустить, что биты в середине определенной части изображения являются коррелированными. Тогда отсутствие такой корреляции может служить сигналом о наличии скрытого сообщения. Задача того, кто встраивает сообщение, заключается в том, чтобы не нарушить статистики контейнера. Отправитель и тот, кто атакует, могут иметь в своем распоряжении разные модели сигналов, тогда в информационно-скрывающем противоборстве победит тот, кто владеет более эффективной (оптимальной) моделью.

Основная цель атаки на стеганографическую систему аналогична цели атак на криптосистему с той лишь разницей, что резко возрастает значимость активных (злонамеренных) атак. Любой контейнер может быть заменен с целью удаления или разрушения скрытого сообщения, независимо от того, существует оно в контейнере или нет. Обнаружение существования скрытых данных ограничивает время на этапе, IX удаления, необходимое для обработки только тех контейнеров, которые содержат скрытую информацию.

Даже при оптимальных условиях для атаки задача извлечения скрытого сообщения из контейнера может оказаться очень сложной. Однозначно утверждать о факте существования скрытой информации можно только после ее выделения в явном виде. Иногда целью стеганографического анализа является не алгоритм вообще, а поиск, например, конкретного стеганоключа, используемого для выбора битов контейнера в стеганопреобразовании.

## 3.3. Основные этапы практического стеганоанализа

Фактически, любое стеганографическое преобразование базируется на двух определяющих принципах [3]:

- в качестве носителя скрытой информации (контейнера) избирается объект, структура которого допускает возможность определенного искажения его собственной информации, сохраняя при этом функциональность объекта;
- уровень внесенных в структуру контейнера искажений должен быть ниже уровня чувствительности средств распознания (в том числе и распознавания органами ощущения человека).

В качестве стеганоконтейнеров, как уже отмечалось выше, могут использоваться практически все известные носители информации, применяемые в современных сетях передачи данных. При этом методы скрытия информации ориентируются, в основном, на внутреннюю структуру контейнера, которая может представлять собой символьные или битовые данные, коэффициенты преобразования Фурье, широкополосное кодирование, коэффициенты уплотнения и т.д. [3].

Скрытие данных в медиасреде требует соблюдения определенных условий при внесении изменений, что необходимо для устранения следов применения операций стеганопреобразования. Например, в случае изображений указанные изменения могут при определенных действиях со стороны нарушителя (как преднамеренных, так и случайного характера) становиться видимыми для человеческого глаза, и, таким образом, явно указывать на использование стеганографических средств. Очевидно, что следы, оставленные последними, могут существенно помочь обнаружить существование скрытого сообщения, таким образом, компрометируя стеганосистему в целом.

Одной из главных задач стеганоанализа является исследование возможных следов применения стеганографических средств и разработка методов, которые позволяли бы обнаруживать факты их использования [3]. Применение конкретного стеганографического преобразования требует от стеганоаналитика индивидуального подхода к его исследованию.

Исследование сообщений, скрытых одним из множества существующих стеганографических методов, или, более точно, подозреваемых в этом отношении, — процесс довольно трудоемкий.

Для успешного проведения стеганоанализа необходимо (но ни в коем случае не достаточно):

- иметь для анализа стеганосредство, с помощью которого осуществляется скрытие сообщения;
- иметь возможность восстанавливать используемые в системе стеганографический и, возможно, криптографический алгоритмы; выполнять их экспертный анализ и разрабатывать алгоритм определения ключей;

- иметь возможность использовать для проведения стеганоанализа вычислительные ресурсы необходимой мощности;
- поддерживать на должном уровне теоретические и практические знания в области компьютерной стеганографии.

Можно выделить следующие несколько направлений практического развития стеганографического анализа [3].

- Разработка вероятностно-статистических методов распознавания, применение элементов искусственного интеллекта для получения оценок надежности стеганографических преобразований, а также при создании детекторов (фильтров) — для анализа информационных потоков с целью обнаружения и перекрытия скрытых каналов связи. В таком случае проверка наличия скрытой информации сводится к определенной оценке с использованием статистических критериев (последовательной корреляции, энтропии изображения, дисперсии младшего бита и т.д.). Разрабатываемые с этой целью средства должны не только обеспечивать низкий уровень погрешности во время распознавания скрытых сообщений (особенно в тех случаях, когда используется предварительное шифрование), но и быть универсальными, то есть должна существовать возможность детектирования сообщений встроенных разными стеганографическими методами.
- Анализ конкретных программных стеганографических средств с целью восстановления алгоритмов и разработки оптимальных методов их исследования. Основная сложность в данном случае заключается в большой трудоемкости, обусловленной необходимостью индивидуального подхода к каждому конкретному алгоритму, реализующему тот или иной метод скрытия информации, а также значительным объемом вычислений, необходимых для восстановления стеганоключей.
- Разработка технологий активных и злонамеренных атак для внесения невосстанавливаемых искажений в предполагаемую стеганограмму с целью спровоцировать ее повторную передачу в другом контейнере, что подтвердило бы факт использования стеганосредств.

## 3.4. Оценка качества стеганоситемы

Создание и эксплуатация надежного стеганографического средства предусматривает наличие определенного инструментария для его контроля и оценки [3]. Количественное оценивание стойкости стеганографической системы защиты к внешним воздействиям представляет собой достаточно сложную задачу, которая обычно на практике реализуется методами системного анализа, математического моделирования или экспериментального исследования.

Как правило, профессионально разработанная стеганосистема обеспечивает трехуровневую модель защиты информации, решающую две основные задачи:

- скрытие самого факта наличия защищаемой информации (первый уровень защиты);
- блокирование несанкционированного доступа к информации, осуществляемое путем избрания соответствующего метода скрытия информации (второй уровень защиты).

Наконец, необходимо принимать во внимание и вероятность существования третьего уровня — предварительной криптографической защиты (шифрования) скрываемой информации.

На рис. 3.2 представлена возможная структура процесса моделирования и оценки устойчивости стеганосистемы [3, 48].

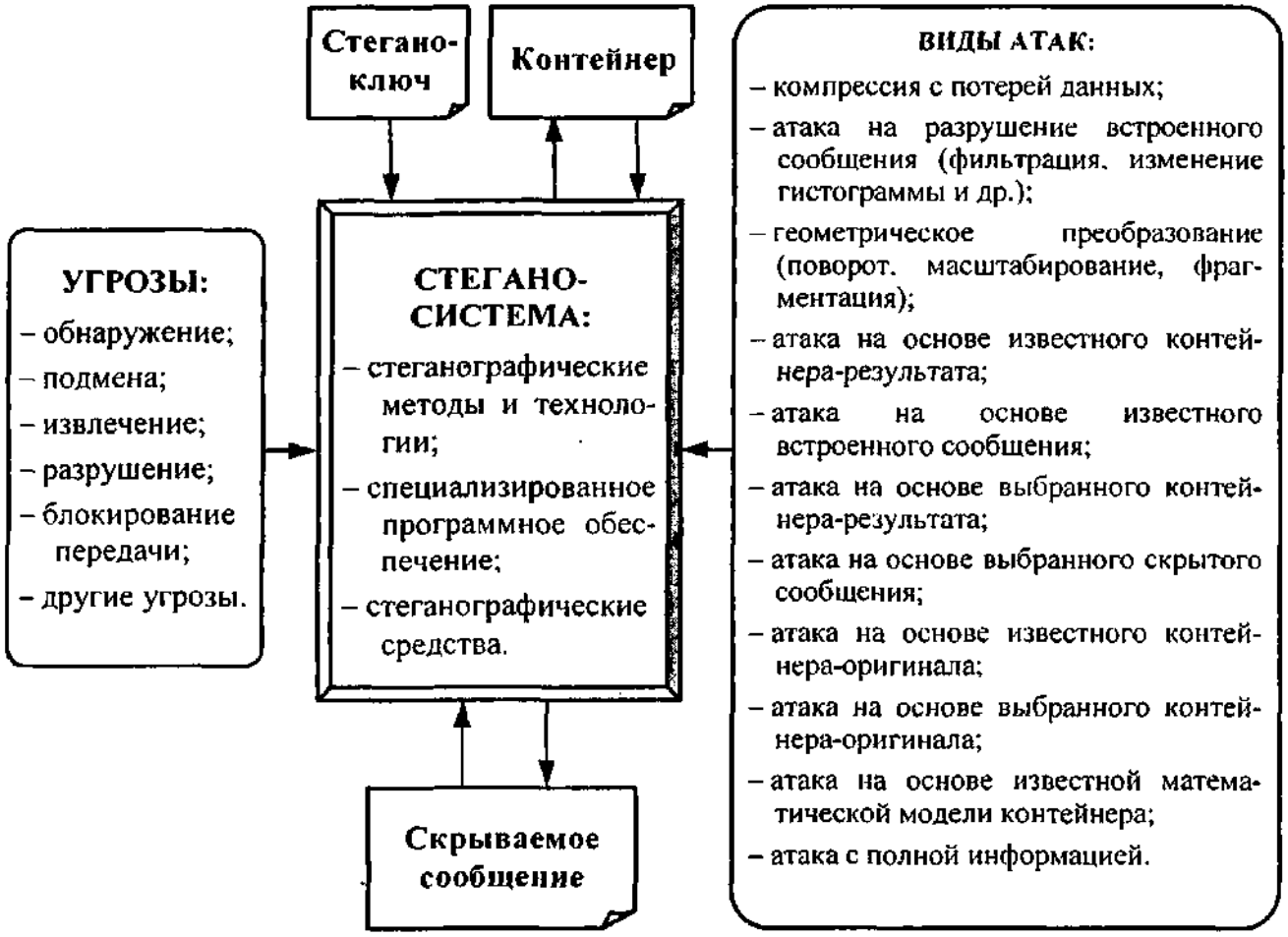



Рис. 3.2. Модель анализа угрозы и оценки устойчивости стеганосистем

Как видим, надежность и время устойчивости стеганосистемы в случае проведения анализа и испытаний определяются возможностями вычислительной системы.

Оценка качества основной характеристики стеганоситемы — *уровня скрытости* — обеспечивается путем проведения аналитических исследований (стеганоанализа) и натурных испытаний [3]. Для оценки качества стеганографического скрытия часто используются известные методы из других областей, в первую очередь — криптоанализа. Поскольку абонент-получатель может восстанавливать скрытую информацию принятого сообщения, то целиком очевидно, что существует некий механизм ее извлечения.

Если нарушитель, выдвигая гипотезы о возможном стеганографическом преобразовании, имеет некоторый инструмент для их проверки, то он имеет шансы подтвердить факт существования скрытой информации, выполнить поиск механизма извлечения секретного сообщения и, наконец, раскрыть содержание сообщения. По этой причине, в первую очередь, для детектирования стеганограмм можно использовать разновидности описанных выше атак на стеганосистему и значительную часть методов криптоанализа.

Достаточно эффективны в некоторых случаях методы оценки уровня скрытости стеганосредств на основании анализа их статистических характеристик. Статистическая теория дает количественные критерии вероятности, что позволяет создавать детекторы, которые будут обнаруживать статистические расхождения между последовательностями. В случае наличия достаточного объема анализируемой информации с достаточно высокой вероятностью можно делать выводы об основных характеристиках последовательности, выделенной для анализа из контейнера.

На начальном этапе анализа рекомендуется воспользоваться традиционными статистическими ($\chi^2$, тесты на запрещенные символы, на длину цикла и т.п.), эмпирическими (проверка частот, серий, интервалов, перестановок; проверка на монотонность, "покер-тестом^, тестом "собирателя купонов") или спектральными тестами [3]. В дальнейшем целесообразно использовать более гибкие методы, которые иногда специально разрабатываются под конкретную задачу.

Для сравнительного оценивания качества стеганографических средств разрабатываются разные показатели, дающие количественные оценки. Больше всего их разработано для стеганометодов, которые работают с изображениями и видео (методов ЦВЗ). Зачастую такие показатели оперируют с изображениями на уровне пикселей, хотя после надлежащей адаптации они применимы и к другим способам описания изображения, а также к аудиоданным. Наиболее популярным показателем при анализе уровня искажений, которые вносятся в контейнер во время скрытия в нем информации, является взятое из радиотехники соотношение "сигнал/шум", вычисленное в децибелах.

В табл. 3.1 представлен ряд показателей, используемых во время оценки искажений, вносимых стеганопреобразованиями в изображение [49-52]. В таблице $\nabla^2 C_{x,y} = C_{x+1,y} + C_{x-1,y} + C_{x,y+1} + C_{x,y-1} - 4 \cdot C_{x,y}$.

**Таблица 3.1. Наиболее распространенные показатели визуального искажения, основанные на анализе пиксельной структуры контейнера**

| **Разностные показатели искажения** | |
|---|---|
| Максимальная разность (Maximum Difference) | $MD = \max_{x,y} \left\| C_{x,y} - S_{x,y} \right\|$ (3.1) |
| Средняя абсолютная разность *(Average Absolute Difference)* | $AD = \frac{1}{XY} \cdot \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\|$ (3.2) |
| Нормированная средняя абсолютная разность *(Normalized Average Absolute Difference)* | $NAD = \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\| \Big/ \sum_{x,y} \left\| C_{x,y} \right\|.$ (3.3) |
| Среднеквадратическая ошибка *(Mean Square Error)* | $MSE = \frac{1}{XY} \cdot \sum \left( C_{x,y} - S_{x,y} \right)^2.$ (3.4) |
| Нормированная среднеквадратическая ошибка *(Normalized Mean Square Error)* | $NMSE = \sum_{x,y} \left( C_{x,y} - S_{x,y} \right)^2 \Big/ \sum_{x,y} \left( C_{x,y} \right)^2$ (3.5) |
| $L^p$-норма *($L^p$-norm)* | $L^p = \left( \frac{1}{XY} \cdot \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\|^p \right)^{1/p}.$ (3.6) |
| Лапласова среднеквадратическая ошибка *(Laplacian Mean Square Error)* | $LMSE = \sum_{x,y} \left( \nabla^2 C_{x,y} - \nabla^2 S_{x,y} \right)^2 \Big/ \sum_{x,y} \left( \nabla^2 C_{x,y} \right)^2.$ (3.7) |

**Таблица 3.1. Окончание**

| | |
|---|---|
| Отношение "сигнал/шум" *(Signal to Noise Ratio)* | $SNR = \sum_{x,y}(C_{x,y})^2 \Big/ \sum_{x,y}(C_{x,y} - S_{x,y})^2.$ (3.8) |
| Максимальное отношение "сигнал/шум" *(Peak Signal to Noise Ratio)* | $PSNR = XY \cdot \max_{x,y}(C_{x,y})^2 \Big/ \sum_{x,y}(C_{x,y} - S_{x,y})^2.$ (3.9) |
| Качество изображения *(Image Fidelity)* | $IF = 1 - \sum_{x,y}(C_{x,y} - S_{x,y})^2 \Big/ \sum_{x,y}(C_{x,y})^2.$ (3.10) |
| **Корреляционные показатели искажения** | |
| Нормированная взаимная корреляция *(Normalized Cross-Correlation)* | $NC = \sum_{x,y} C_{x,y} \cdot S_{x,y} \Big/ \sum_{x,y}(C_{x,y})^2.$ (3.11) |
| Качество корреляции *(Correlation Quality)* | $CQ = \sum_{x,y} C_{x,y} \cdot S_{x,y} \Big/ \sum_{x,y} C_{x,y}.$ (3.12) |
| **Другие показатели** | |
| Структурное содержание *(Structural Content)* | $SC = \sum_{x,y}(C_{x,y})^2 \Big/ \sum_{x,y}(S_{x,y})^2.$ (3.13) |
| Общее сигма-отношение "сигнал/шум" *(Global Sigma Signal to Noise Ratio)* | $GSSNR = \sum_b \sigma_b^2 \Big/ \sum_b (\sigma_b - \tilde{\sigma}_b)^2,$ (3.14)<br>где $\sigma_b = \sqrt{\frac{1}{n} \cdot \sum_{\text{блок } b}(C_{x,y})^2 - \left(\frac{1}{n} \cdot \sum_{\text{блок } b} C_{x,y}\right)^2}.$ |
| Сигма-отношение "сигнал/шум" *(Sigma Signal to Noise Ratio)* | $SSNR' = \frac{1}{n} \cdot \sum_b SSNR_b,$ (3.15)<br>где $SSNR_b = 10 \cdot \lg\left[\frac{\sigma_b^2}{(\sigma_b - \tilde{\sigma}_b)^2}\right].$ |
| Нормированное отношение "сигма/ ошибка" *(Normalized Sigma to Error Ratio)* | $NSER = \frac{1}{\max(SER)} \cdot \sum_b SER_b,$ (3.16)<br>где $SER_b = \sigma_b^2 \Big/ \left[\frac{1}{n} \cdot \sum_{\text{блок } b}(C_{x,y} - S_{x,y})^2\right].$ |
| Подобие гистограмм *(Histogram Similarity)* | $HS = \sum_{c=0}^{255} \lvert f_C(c) - f_S(c) \rvert,$ (3.17)<br>где $f_C(c)$ — относительная частота градации цвета $c$ в изображении с 256 уровнями цветов. |

Большинство показателей искажения или критериев качества, которые используются при визуальной обработке информации, относятся к группе *разностных показателей искажения.* Эти показатели базируются на отличии между контейнером-оригиналом (неискаженный сигнал) и контейнером-результатом (искаженный сигнал). Ко второй группе относятся показатели, основанные на корреляции между оригинальным и искаженным сигналами (так называемые *корреляционные показатели* ***искажения).***

В представленных соотношениях через $C_{x,y}$ обозначается пиксель пустого контейнера с координатами $(x, y)$, а через $S_{x,y}$ — соответствующий пиксель заполненно-

го контейнера. В параметрах ***GSSNR, SSNR*** и ***SER*** анализированное изображение предварительно разбивается на *N* блоков по ***n*** пикселей размером ***X*** × ***Y***, где *X* и *Y*— соответственно, количество строк и столбцов в блоке (например, блок 8x8). Более детальное описание показателей можно получить, в частности, из [52].

Рассмотренные выше показатели базируются на анализе отдельных элементов сигнала (в данном случае — пикселей изображения). Слабые места таких показателей известны уже на протяжении длительного времени (например, отсутствие корреляции разностных показателей искажения со зрением человека). В последнее десятилетие все больше исследований направлено на поиски такого показателя искажения, который был бы адаптирован к человеческой зрительной или слуховой системе путем учета разнообразных влияний [53-56]. Рассмотрим показатель искажения, который предложили в свое время Фарелл (J.E. Farrell) и Ван ден Бранден Ламбрехт( C.J. van den Branden Lambrecht) [55].

Степень воспринимаемого человеком качества оперирует чувствительностью к контрасту и явлением маскировки системой восприятия человека и базируется на многоканальной модели человеческого пространственного зрения.

Вычисление данного показателя требует следующих действий:

- проведение крупношаговой сегментации изображения;
- разложение ошибки кодирования и первичного изображения на перцепционные (относящиеся к восприятию органами чувств) компоненты, используя гребенчатые фильтры;
- вычисление порога обнаружения для каждого пикселя, используя первичное изображение как маску;
- распределение фильтрованной ошибки с помощью порога принятия решения, объединение по всем цветовым каналам.

Единица измерения показателя определяется как единица *превышения порога,* что подразумевает под собой только значимое (заметное) отличие (Just Noticeable Difference — JND). Общий показатель, скрытое максимальное соотношение "сигнал/шум"* (Masked Peak Signal to Noise Ratio — MPSNR):

$$MPSNR = 10 \cdot \lg\left(\frac{255^2}{\varepsilon^2}\right) \tag{3.18}$$

где ε — вычисленное искажение. Поскольку показатель качества не соответствует смыслу, заложенному в понятие "децибел", его называют *визуальным* или *зрительным децибелом* (ВдБ).

В большинстве случаев более полезна нормализованная оценка качества. В [49] предлагается использовать оценку качества *Q* в соответствии с рекомендацией сектора радиосвязи МСЭ — ITU-R Rec. 500:

$$Q = \frac{5}{1 + N \cdot \varepsilon} \tag{3.19}$$

где ε — вычисленное искажение; *N*— нормирующий коэффициент, который зачастую выбирается таким, чтобы характеристика искажения отображала соответствующую качественную оценку.

В табл. 3.2 представлены оценки и соответствующие им зрительное восприятие внесенных искажений и качество подвергнутых обработке изображений." Такая оценка имеет несколько преимуществ, а именно, отсутствие разрушения неискаженных изображений.

Таблица 3.2. ITU-R Rec. 500. Оценка качества по шкале от 1 до 5

| Оценка | Искажение | Качество |
|---|---|---|
| 5 | Незаметное | Отличное |
| 4 | Заметное, не раздражающее | Хорошее |
| 3 | Несущественно раздражающее | Удовлетворительное |
| 2 | Раздражающее | Неудовлетворительное |
| 1 | Чрезвычайно раздражающее | Крайне неудовлетворительное |

## 3.5. Абсолютно надежная стеганосистема

В [3, 15] представлено формальное теоретико-информационное определение устойчивости стеганосистемы относительно пассивных атак. Главная идея базируется на случайности избрания контейнера *c* из множества C с вероятностью $P_C$

Встраивание в контейнер секретного сообщения можно описать как функцию, определенную на множестве C. Пусть $P_C$ — вероятность формирования стеганосистемы $E(c, m, k)$ на множестве *S* всех возможных стеганограмм, полученных с помощью стеганосистемы. Если контейнер *c* никогда не используется для получения стеганограмм, то $P_S(c) = 0$. Для вычисления вероятности $P_S$ необходимо учитывать распределение вероятностей на множестве ключей *K* и множестве сообщений *M*.

Определим на множестве *Q* такое соотношение для относительной энтропии, с помощью которого можно измерить неэффективность принятия неверной гипотезы о распределении *P*\ в случае истинного распределения $P_0$:

$$D(P_0 \| P_1) = \sum_{q \in Q} P_0(q) \cdot \log_2 \left( \frac{P_0(q)}{P_1(q)} \right), \tag{3.20}$$

где выражение $\log_2(\cdot)$ является алгоритмическим отношением правдоподобия.

Относительная энтропия между двумя распределениями всегда неотрицательна и равна 0 только в случае тождественности данных распределений. Таким образом. для стеганопреобразования можно получить некоторую оценку.

Представим определение надежности стеганосистемы в терминах относительной энтропии.

**Определение 3.1**

Пусть Σ — стеганографическая система; *P$*— распределение вероятностей пере- I дачи каналом связи стеганограмм; $P_c$— распределение вероятностей передачи каналом связи пустых контейнеров. Система Σ называется *p-надежной* к пассив- | ным атакам, если $D(P_C \| P_S) \le \rho$, и является *абсолютно надежной,* если ρ = 0.

Как уже указывалось, соотношение $D(P_C \| P_S)$ равно нулю только в том случае, когда оба распределения вероятностей равны друг другу. Следовательно, стеганосистема Σ является ***теоретически абсолютно надежной,*** если процесс встраивания секретного сообщения в контейнер не изменяет распределение $P_C$. Абсолютно безопасная система может быть создана, например, на основании одноразовой гаммы [3].

На основании сказанного, формулируется следующая теорема.

**Теорема 3.1**

Существует абсолютно надежная стеганосистема.

Доказательство

В [15] проведено конструктивное доказательство данного утверждения. Пусть контейнер *C* представляет собой равномерно распределенную $n$-битную последовательность для некоторого положительного *n*. Отправитель с помощью генератора ключа получает равномерно распределенный $n$-битный ключ *K*. Считается, что функция встраивания заключается в побитовом сложении по модулю 2 (операция XOR) $n$-битного секретного сообщения (в роли которого в данном случае выступает собственно контейнер *C)* с ключом $K$ : $S = C \oplus K$. Получатель декодирует полученную последовательность повторным применением операции XOR : $C = S \oplus K$. Совершенно очевидно, что результирующая стеганограмма *S* также будет представлять собой распределенную я-битную последовательность. Следовательно, $P_C \sim P_S$, откуда $D(P_C \| P_S) = 0$. Теорема доказана.

## 3.6. Устойчивость стеганосистем к пассивным атакам

Пассивный нарушитель пытается найти ответ на вопрос: содержит перехваченный им контейнер скрытую информацию или нет? Для этого ему необходимо провести оценку неоднородности некоторых параметров контейнера, обнаружить в нем "подозрительные" участки, завышенное зашумление и другие следы присутствия скрытых сообщений. Эта задача может быть формализована в виде проблемы проверки статистических гипотез [5]. С этой целью вводят тестовую функцию стеганодекодера, которая, в зависимости от типа последнего, может выдавать двухразрядные (в более сложном случае — μ-разрядные) решения о наличии/отсутствии встроенного сообщения:

$$D: C \to \{0;1\}, \tag{3.21}$$

$$D(c) = \begin{cases} 1, \text{ если контейнер содержит скрытое сообщение;} \\ 0, \text{ если скрытое сообщение в контейнере отсутствует.} \end{cases}$$

С помощью данной функции нарушитель способен оценивать сообщения, перехваченные им в несекретном канале. В качестве детектора скрытых сообщений зачастую используют корреляционный приемник, изображенный на рис. 3.3.

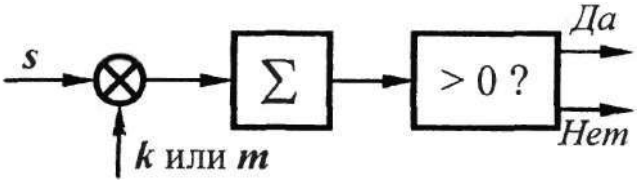



Рис. 3.3. Корреляционный детектор скрытых сообщений

Пусть вследствие скрытия в изображении-контейнере сообщения, у некоторой части пикселей значение яркости было увеличено на 1, а у других осталось неизменным, или же было уменьшено на 1. Тогда s = *c* + *tm,* где $m = E(c, k)$. Коррелятор детектора вычисляет величину $s \cdot m = (c + m) \cdot m = c \cdot m + m \cdot m$. Поскольку *m* принимает значение $\pm 1$, то $\Sigma\, c \cdot m \to 0$, а $m \cdot m$ всегда будет положительным, поэтому *s m* будет очень близким к $m \cdot m$. Тогда можно воспользоваться сведениями из теории связи и записать вероятность ошибочного обнаружения наличия скрытого сообщения как дополнительную (комплементарную) функцию ошибок от корня квадратного из от-

ношения $m \cdot m$ ("энергии сигнала") к дисперсии значений пикселей яркости ("энергия шума") [5].

В детекторе возможно возникновение двух типов ошибок. Существует вероятность того, что при анализе контейнера детектор не обнаружит присутствующего в нем скрытого сообщения (β-вероятность пропуска цели или так называемая *ошибка 2-го рода),* а также вероятность ошибочного обнаружения скрытого сообщения в пустом контейнере (α-вероятность ложной тревоги или так называемая *ошибка 1-го рода).* Уменьшение одной вероятности приводит к возрастанию другой. На практике пытаются максимизировать для пассивного атакующего ошибку 2-го рода, идеальная же стеганосистема должна обеспечивать ошибку 2-го рода $\beta = 1$. Ниже будет показано, что все абсолютно надежные стеганосистемы имеют это свойство (при условии, что атакующий совершает ошибку 1-го рода с вероятностью $\alpha = 0$).

Для ρ-надежных стеганографических систем вероятность $\alpha$ и $\beta$ связаны между собой в соответствии с представленной ниже теоремой [15].

**Теорема 3.2**

Пусть Σ — стеганографическая система, которая является ρ-надежной против пассивных атак. Тогда вероятность β того, что нарушитель не обнаружит скрытое сообщение, и вероятность α того, что он ошибочно обнаружит несуществующее скрытое сообщение, удовлетворяют соотношению $d(\alpha, \beta) \le \rho$, где $d(\alpha, \beta)$ — относительная двоичная энтропия, которая определяется как

$$d(\alpha,\beta) = \alpha \cdot \log_2 \frac{\alpha}{1-\beta} + (1-\alpha) \cdot \log_2 \frac{1-\alpha}{\beta}. \qquad (3.22)$$

В частности, если $\alpha = 0$, то $\beta \ge 2^{-\rho}$.

Доказательство

В случае, когда контейнеры не содержат скрытого сообщения, то их распределение вероятностей соответствует $P_C$. Рассмотрим случайную величину $f(C)$ и вычислим ее вероятность $\pi_C$. Если $f(C) = 1$ (пустой контейнер принимается за стеганограмму), нарушитель совершает ошибку 1-го рода. Таким образом, $\pi_C(1) = \alpha$ и $\pi_C(0) = 1 - \alpha$.

Если контейнер содержит скрытое сообщение, то распределение вероятностей соответствует $P_S$. Вычислим вероятность $\pi_S$ для $f(S)$. В случае, когда $f(S) = 0$ (стеганограмма принимается за пустой контейнер), нарушитель совершает ошибку 2-го рода. Следовательно, $\pi_S(0) = \beta$ и $\pi_S(1) = 1 - \beta$. Относительная энтропия $D(\pi_C \| \pi_S)$ в соответствии с (3.20) может быть выражена следующим образом:

$$D(\pi_C \| \pi_S) = \sum_{q \in \{0;1\}} \pi_C(q) \cdot \log_2 \frac{\pi_C(q)}{\pi_S(q)} =$$

$$= \alpha \cdot \log_2 \frac{\alpha}{1-\beta} + (1-\alpha) \cdot \log_2 \frac{1-\alpha}{\beta} = d(\alpha,\beta).$$

Отметим следующее свойство функции относительной энтропии: детерминированная обработка не может увеличить относительную энтропию между двумя распределениями. Пусть $Q_0$ и $Q_1$ — две случайные величины, определенные на

множестве наблюдений $Q$ с соответствующими распределениями вероятностей $P_{Q_0}$ и $P_{Q_1}$, а $f$ — произвольная функция отображения, преобразующая множество наблюдений $Q$ во множество наблюдений $T$ ($f: Q \to T$). Тогда справедливо следующее выражение:

$$D\left(P_{T_0} \| P_{T_1}\right) \le D\left(P_{Q_0} \| P_{Q_1}\right)$$

где через $T_0$ и $T_1$ ($T_0, T_1 \in T$) соответственно помечены случайные величины $f(Q_0)$ и $f(Q_1)$. Таким образом,

$$d(\alpha,\beta) = D\left(\pi_C \| \pi_S\right) \le D\left(P_C \| P_S\right) \le \rho$$

Учитывая, что

$$\lim_{\alpha \to 0} \log_2[\alpha/(1-\beta)] = 0$$

получаем:

$$d(0,\beta) = \log_2(1/\beta)$$

Следовательно, если $\alpha = 0$, то $\beta \ge 2^{-\rho}$. Теорема доказана.

Итак, для $\rho$-надежной стеганосистемы с $\alpha = 0$, необходимо обеспечить, чтобы $\rho \to 0$. При этом вероятность $\beta \to 1$, что эквивалентно невозможности выявления нарушителем скрытого в контейнере сообщения.

## 3.7. Активные и злонамеренные атаки

Даже при условии невозможности выделения и чтения скрытого сообщения, факт наличия последнего можно относительно легко обнаружить. Еще более простой является операция уничтожения данного сообщения. Например, если сообщение скрыто в файле BMP-формата методом замены палитры, то воздействие на этот файл случайной заменой цветов в палитре сделает сообщение неизвлекаемым, другими словами — уничтожит его.

Во время проектирования и исследования стеганографических систем особое внимание должно быть уделено изучению влияния на них активных и злонамеренных атак [3]. *Активные атаки* способны изменить контейнер во время связи: нарушитель может перехватить стеганограмму, которая была послана передающей стороной к принимающей, изменить ее (путем определенной обработки контейнера) и отправить результат принимающей стороне. В данном случае предполагается, что при активной атаке невозможно полностью заменить контейнер и его семантику, а можно только внести незначительные изменения таким образом, чтобы оригинал и измененный контейнер остались визуально и семантически подобными. *Злонамеренными атаками* являются такие, при которых существует возможность подменить сообщение другим. При этом организовывается фальсифицированный стеганографический обмен под именем одного из партнеров связи.

Стеганографические системы чрезвычайно чувствительны к модификации контейнера (например, для изображения — это сглаживание и фильтрация, для звука — фильтрация и переквантование отсчетов). Так, простое сжатие с потерями может привести к полной потере информации, поскольку при этом извлекаются незначительные компоненты сигнала и, этим самым, уничтожается скрытая в них секретная информация. Во время активных атак, когда нет возможности извлечь скрытую ин-

формацию или доказать ее существование, ее можно уничтожить простым добавлением в стеганограмму случайных помех. В случае цифровых изображений атакующий может применить распространенные методы обработки изображений, в том числе изменить его формат.

В современных компьютерных системах реализуются стеганографические преобразования с высокой избыточностью, устойчивые к трансформации контейнера (поворот, масштабирование, печать с последующим сканированием и т.д.), поэтому одно из важных требований к прикладной стеганосистеме — это обеспечение устойчивости к случайным или умышленным атакам.

## 3.8. Устойчивость стеганографической системы к активным атакам

Исходя из рассмотренных выше особенностей атак на стеганосистемы, можно сделать вывод, что противодействие статистическому стеганоанализу должно заключаться в построении математических моделей сигналов-контейнеров, поиске на их основании "разрешенных" для модификации областей и встраивании в них скрываемой информации, статистика которой не отличается от статистики контейнера. Такая неотличимость определяет устойчивость стеганосистемы.

Как и для криптографических систем защиты информации, безопасность стеганосистем описывается и оценивается их устойчивостью (стеганографической устойчивостью), однако определения устойчивости и взлома данных систем различны. В криптографии система защиты информации является устойчивой, если, владея перехваченной криптограммой, нарушитель не способен извлечь содержащуюся в ней информацию.

Неформально определим, что стеганосистема является *устойчивой,* если нарушитель, наблюдая за информационным обменом между отправителем и получателем, неспособен обнаружить, что под прикрытием контейнеров передаются скрываемые сообщения, и тем более узнать о содержании последних. В более широком смысле, под устойчивостью стеганосистем понимается их способность скрывать от квалифицированного нарушителя факт скрытой передачи данных, способность противостоять попыткам нарушителя разрушить, исказить, удалить скрыто передаваемые сообщения, а также возможность подтвердить или опровергнуть аутентичность скрыто передаваемой информации.

Стеганографическая система является устойчивой к активным атакам, если скрываемая с ее помощью информация не может быть заменена без значительных изменений контейнера, в результате которых последний потеряет свою функциональность [3].

Определение 3.2

Пусть Σ — стеганографическая система и φ — класс отображений $C \to S$. Тогда система Σ будет φ-устойчивой, если в случае стеганосистем с секретным ключом для всех $f \in \varphi$ справедливо

$$D\{f[E(c, m, k), k]\} = D[E(c, m, k), k] = m, \qquad (3.23)$$

а в случае бесключевых стеганосистем, независимо от выбора $m \in M$, $c \in C$ и $k \in K$:

$$D\{f[E(c, m)]\} = D[E(c, m)] = m. \qquad (3.24)$$

Очевидно, что существует и обратная взаимосвязь между надежностью стеганосистемы и ее устойчивостью: чем более устойчивой к модификации контейнера будет стеганосистема, тем она будет менее надежной, поскольку устойчивость может быть достигнута помехоустойчивым кодированием, что может привести к существенным искажениям контейнера и, возможно, к изменению вероятности $P_s$.

Многие стеганосистемы устойчивы только к определенному классу отображений (уплотнение с потерями, геометрические преобразования, фильтрация, переквантование отсчетов, аддитивный белый шум, преобразования ЦАП→АЦП и т.д.). Идеальная стеганосистема должна быть устойчивой ко всем отображениям типа "сохранение λ-подобия", то есть, отображением вида $f: C \to S$ со свойством $sim[c, f(c)] \to \lambda$ и $\lambda \approx 1$ [3]. Однако такие системы сложны в проектировании и, из-за необходимости дополнительного применения помехоустойчивого кодирования, имеют слишком низкую пропускную способность. С другой стороны, система является "λ-слабой", если для каждого контейнера существует такое отображение "сохранения λ-подобия", что скрытая информация будет невосстанавливаемой с точки зрения соотношений (3.23) или (3.24).

В общем случае, существует два подхода к созданию устойчивых стеганосистем [3]:

- предусматривая возможные атаки на стеганограммы со стороны нарушителей, стеганографическое преобразование изначально проектируется устойчивым к уничтожению скрытых данных определенным классом модификаций;
- используются преобразования, которые имеют свойство обратимости к возможным модификациям с целью восстановления начального вида стеганограммы. Так в [14] предложен метод "афинного кодирования" для противодействия афинным преобразованиям изображения. При этом предусматривается оценка параметров преобразований, измерение изменений формы, размеров некоторых кодированных образов.

Устойчивые алгоритмы должны скрывать данные в наиболее существенных фрагментах контейнера, поскольку информация, кодируемая в шумовом компоненте, легко может быть извлечена или разрушена. Например, известно [57], что стеганографические преобразования, которые работают с частотной областью контейнера, преимущественно более устойчивы к модификациям по сравнению с алгоритмами, работающими в пространственной (для изображения) или временной (для звука) областях. Используя эти свойства, можно создать устойчивые стеганосистемы, которые, например, будут сохранять скрываемую информацию в коэффициентах дискретного косинусного преобразования (ДКП) изображения.

## 3.9. Сознательно открытый стеганографический канал

Выше были рассмотрены модели стеганосистем, в которых используется секретный ключ (private-key), разделяемый между определенными участниками стеганографического обмена. Подобные модели являются ограниченными для пассивного нарушителя, который, тем не менее, способен обнаружить факт передачи скрытого сообщения и при определенных обстоятельствах узнать о его содержании. Другое дело, если нарушитель является активным или злонамеренным. Тогда он может не только вносить помехи в стеганоканал, но и полностью имитировать отправителя. Поскольку в большинстве случаев априорная информация об отправителе у получателя отсутствует, он не способен отличить фальсификацию, поэтому скрытая передача данных при наличии активного нарушителя — это намного более сложная проблема по сравнению с фактом присутствия пассивного нарушителя.

В работах [5, 12] представлен протокол, позволяющий решить эту задачу. Данный протокол основан на введении к рассмотрению канала с исключительно малой пропускной способностью — сознательно открытого (supraliminal) канала. Такой канал образуется за счет встраивания скрываемых данных в наиболее важные характеристики контейнера, искажение которых приведет к полной деградации последнего. Другими словами, информация встраивается в контейнер таким образом, что ее видно, но невозможно изменить без существенных изменений характерных свойств контейнера.

Очевидна и сфера использования указанного протокола — концепция открытого стеганоканала используется преимущественно для встраивания ЦВЗ. Рассмотрим принцип его использования [5].

Предположим, при активной атаке нарушителю удается внести только незначительные изменения в пересылаемый контейнер. Следовательно, некоторая информация, которая специфична для конкретного контейнера, сохранится, поскольку ее нельзя удалить без существенного изменения семантики контейнера. Дело в том, что нарушитель во многих случаях не может вносить помехи в стеганоканал, значительные настолько, чтобы передаваемая информация была полностью искажена. Не может по причинам не технического характера, а из юридических или других соображений. Поэтому, если секретное сообщение встроить в существенные фрагменты контейнера, то его можно передавать между абонентами с высокой степенью целостности даже при наличии активных помех.

Встраивание информации в наиболее значимые элементы контейнера — основной принцип применения ЦВЗ. Отличие сознательно создаваемого открытого канала заключается в том, что для встраивания и извлечения данных не нужен секретный ключ. Место размещения скрываемых битов общеизвестно, но их невозможно удалить без заметного разрушения контейнера. Кроме того, ЦВЗ может не содержать в себе никакой осмысленной информации (например, быть функцией самого изображения). В случае же сознательно открытого канала, наоборот, контейнер может быть функцией скрываемого короткого сообщения.

Допустим, что каждому контейнеру соответствует некий шаблон, в котором формально описаны все характерные особенности контейнера [3]. Пусть $S$ — множество всех шаблонов и $f: S \to \{0; 1\}^N$ - функция шаблонов. Для того чтобы передавать битовую строку данных $m \in \{0; 1\}^N$, передающая сторона выбирает из множества $S$ некий шаблон $s \in f^{-1}(m)$ и пересылает открытым каналом контейнер, которому соответствует этот шаблон. Пассивный нарушитель может подозревать о существовании скрытого обмена в шумовом компоненте контейнера и с целью разрушения секретного сообщения может несколько изменить стеганограмму, однако при этом он не в состоянии изменить шаблон контейнера. В свою очередь, принимающая сторона может восстановить шаблон $s$ из принятой стеганограммы (возможно, модифицированной пассивным нарушителем) и извлечь данные $m$ с помощью функции $f$.

Использовать открытый стеганоканал для пересылки битовой строки $m$ с явным содержанием нецелесообразно, поскольку пассивный нарушитель с помощью открытой функции $f$ может легко восстановить встроенную информацию. Однако, если сообщение $m$ является случайным секретным ключом или выглядит как случайный шифротекст (то есть, предварительно защищено криптографически), то у нарушителя не будет оснований для подозрений и доказательств (если, конечно, он не способен взломать криптосистему), что переданная информация является чем-то более существенным, чем случайная битовая строка.

Для практической реализации данного протокола необходимо создать ряд условий. Очевидно, что основные трудности заключаются в формировании контейнера, тогда как работа по извлечению данных может быть легко автоматизирована. Для практического же применения сознательно открытого канала должны быть автоматизированы обе операции.

Во-первых, должна существовать возможность создания для любого шаблона такого контейнера, небольшие изменения которого при активных атаках не будут приводить к изменениям скрытых данных. При этом у пассивного нарушителя не должно существовать возможности путем манипуляций со стеганограммой изменить шаблон *s* и привести его к такому виду $s'$, что $f(s) \neq f(s')$.

Во-вторых, должна существовать возможность формирования шаблона для каждого полученного контейнера. Кроме того, функция / должна быть общедоступной, а / и $f^{-1}$ — вычисляемыми. Извлечение сообщения из пустого контейнера должно возвращать случайную строку данных. Следовательно, единственное отличие между заполненным и пустым контейнерами заключается в том, что строка $f(s)$ имеет осмысленное значение.

Описанная схема может быть также применена для скрытого обмена ключами. Протокол обмена следующий [12].

1. Передающая сторона генерирует пару открытого $(E)$ и секретного $(D)$ ключей; вычисляет представительное описание (шаблон) контейнера, соответствующее ключу $E$: $s_E \in f^{-1}(E)$; генерирует контейнер, соответствующий $s_E$, и отсылает его принимающей стороне.
2. Принимающая сторона извлекает из принятого контейнера открытый ключ *E* передающей стороны: $E \in f(s_E)$; генерирует свой секретный ключ *K;* шифрует его с помощью открытого ключа *E;* находит соответствующее полученной последовательности $K_E$ описание (шаблон) контейнера: $s_{KE} \in f^{-1}(K_E)$; генерирует контейнер, соответствующий $s_{KE}$, и отсылает его передающей стороне.
3. Передающая сторона восстанавливает зашифрованный секретный ключ: $K_E \in f(s_{KE})$ и расшифровывает его, используя свой секретный ключ *D.*

Теперь стороны могут обмениваться сообщениями, встроенными в контейнер с использованием секретного ключа *K.* Нарушитель в результате перехвата канала может получить открытый ключ передающей стороны и зашифрованный этим ключом секретный ключ принимающей стороны. Значение последнего без знания секретного ключа передающей стороны будет оставаться для него неизвестным.

Следует отметить, что главная проблема схемы открытого стеганографического канала заключается в эффективной реализации функции/. Кроме того, такой канал не подходит для скрытой передачи сообщений большого объема, поскольку имеет низкую пропускную способность и является открытым для нарушителя.

Фон Ан (L. von Ahn) и Хоппер (N.J. Hopper) [58] формализовали стеганографию с открытым ключом в случае пассивного нарушителя, а также создали ограниченную модель при наличии нарушителя, который осуществляет активную атаку на стеганоканал. На их взгляд, необходимо обеспечивать устойчивость против атак конкретных (заранее определенных) нарушителей в тех случаях, когда получатель должен быть уверен в аутентичности отправителя. Это, по мнению авторов [59], является ограничением модели по сравнению с принципами, заложенными в процесс обмена с открытым ключом.

В своей работе они предлагают комплексную теоретическую модель протокола стеганографического обмена с открытым ключом в случае активных атак, причем

лица, принимающие в этом участие, априори не нуждаются в разделении секретной информации, а нарушитель может влиять на канал и осуществлять так называемую *адаптивную к контейнерам атаку*. Такой вид атаки представляется наиболее обобщающим по сравнению со стеганосистемами, протоколы которых построены с использованием открытого ключа. Это позволяет нарушителю отсылать принимающей стороне произвольную последовательность адаптивно выбранных встроенных в контейнер сообщений и изучать интерпретацию ею каждого из сообщений; то есть, рассматривает ли получатель сообщения как пустой контейнер или же как стеганограмму (с извлечением в последнем случае встроенных данных).

Описанная в [59] модель построена на предположении, что стеганосистема с открытым ключом по своей сути является криптографической системой с открытым ключом с дополнительным необходимым условием, что результат ее работы (стеганограмма) должен соответствовать распределению использованного при этом контейнера.

## 3.10. Выводы

В данном разделе путем исследования известных публикаций отечественных и зарубежных авторов выполнено системное изложение вопросов надежности и устойчивости произвольной стеганографической системы по отношению к видам совершаемых на нее атак. Последние были разделены в соответствии с классификацией типов нарушителей на пассивных, активных и злонамеренных.

Проведение аналогии между стегано- и криптоанализом позволило выделить как общие, так и характерные только для стеганосистем виды атак (атака на основании известного пустого контейнера, атака на основании выбранного пустого контейнера, атака на основании известной математической модели контейнера или его части).

Был выполнен обзор ряда публикаций, посвященных показателям, используемым с целью оценки искажений, вносимых стеганопреобразованиями в пиксельную структуру контейнера.

# Глава 4

# Пропускная способность каналов передачи скрываемых данных

## 4.1. Понятие пропускной способности

Для разрабатываемых или исследуемых стеганографических систем важно определить, насколько большой может быть пропускная способность создаваемых при этом каналов передачи скрываемых данных (КПСД), и как она будет зависеть от других характеристик стеганосистем и условий их использования. Под *пропускной способностью* каналов передачи скрываемых данных или просто *скрытой пропускной способностью* (СПС) понимают максимальное количество информации, которая может быть встроена в один элемент (например, пиксель или временной отсчет) контейнера. Обязательным условием при этом является безошибочность передачи скрываемых данных получателю, а также их защищенность от таких атак нарушителя как попытка выявления факта наличия канала скрытой связи, получение содержания скрытых сообщений, умышленное введение сфальсифицированных данных или же разрушение встроенной в контейнер информации [5].

Канал скрытой связи (КСС) образуется внутри канала открытой связи (КОС), для которого Шеннон (CE. Shannon) в своих работах по теории информации определил пропускную способность [45, 60, 70]. Пропускная способность КОС определяется как количество информации, которую потенциально можно передать без ошибок за одно использование канала. При этом не выдвигается ни одного требования к защищенности от атак организованного нарушителя. Поэтому будет совершенно логично предположить, что скрытая пропускная способность КСС, в котором за одно использование канала передается один элемент контейнера с вложенной в него скрытой информацией, ни в коем случае не может быть больше пропускной способности КОС.

На сегодняшний день наметились разные, иногда диаметрально противоположные подходы к определению количества информации, которая подлежит защите от разнообразных атак нарушителя с помощью стеганографических методов. Эти расхождения, как указывается в [5], обусловлены различиями в целях защиты информации, видах нарушителей, их возможностях и реализованных атаках на стеганосистемы, видах используемых контейнеров и скрываемых сообщений и многими другими факторами.

В той же работе предлагается выполнить оценку величины ПС КПСД методами теории информации для разных стеганосистем. Теоретико-информационные методы позволяют получить строгие оценки количества скрываемой информации, которые совершенно правомерно могут быть использованы как теоретически достижимые предельные скорости передачи скрытой информации для стеганосистем, не учитывая принципы, заложенные в основу их построения.

Предлагается рассмотреть два основных подхода к оценке пропускной способности КПСД. Первый из них, развитый в работах [61, 62], ориентирован на стеганографические системы, в которых сообщения, подлежащие защите, должны быть безошибочно переданы в условиях активного противодействия нарушителя. Этот подход описывает сценарий скрытия так называемых безызбыточных сообщений в данных контейнера, и, что самое главное, позволяет учитывать тот факт, что кроме искажений структуры контейнера при встраивании в него скрываемых данных, возможны его умышленные искажения со стороны нарушителя. Кроме того, существует еще и вероятность искажений случайного характера, вызванных неумышленными помехами в канале связи.

Нарушитель, кроме пассивных действий анализа, может использовать и активные действия (активный нарушитель). Целью активного нарушителя является разрушение скрытой информации. Такая постановка задачи информационного скрытия является характерной, например, для систем ЦВЗ.

В [5] задача информационного скрытия сформулирована как задача безошибочной передачи скрываемой информации при влиянии случайных и умышленных помех, а также определена максимальная скорость безошибочной передачи при разных стратегиях действий отправителя и атакующего. Предложенный подход определяет *теоретически достижимую* скорость достоверной передачи скрываемых сообщений, хотя в явном виде и не оценивает защищенность последних от выявления факта их существования. Однако для ряда стеганосистем не нужно скрывать факт использования стеганографической защиты: владелец авторских или имущественных прав на медиаконтейнер, который защищен ЦВЗ, как правило, открыто сообщает о применении данной системы защиты. В рассмотренном подходе исследуются условия, при которых скрываемая информация гарантированно передается в условиях произвольных попыток нарушителя относительно ее разрушения.

Знание параметров стеганосистемы и возможных стратегий действий передающей стороны не должно позволить нарушителю оптимизировать разрушительное влияние и оценить его эффективность. Особенностью таких стеганосистем является, во-первых, то, что разрушительное влияние происходит только в момент передачи скрытых данных и должно осуществляться в режиме реального времени. Во-вторых, существует априорная неосведомленность законного получателя относительно скрыто передаваемой ему информации. В-третьих, нарушитель в подавляющем большинстве случаев не способен достоверно оценить эффективность своих действий.

Другая ситуация возникает при попытке активного нарушителя разрушить ЦВЗ с целью присвоить себе контейнер (права на него). Нарушитель может как угодно долго осуществлять разрушительное влияние, выбирая такую оптимальную стратегию, при которой, разрушив ЦВЗ, он сохранит необходимое ему качество контейнера. При этом он заранее знает о существовании скрытой информации, и, используя общеизвестный детектор (см. рис. 3.3.), способен оценить эффективность своих атак на ЦВЗ.

Другой подход, который предлагают, например, авторы работ [19, 44, 46], дает оценку скрытой пропускной способности непосредственно в процессе встраивания скрываемых сообщений в избыточные данные контейнера. Такой подход учитывает, что контейнеры формируются реальными избыточными источниками с существенной памятью, такими как источники изображений или аудиосигналов. В этом случае оценка ПС зависит от характеристик замаскированности скрытого канала.

Такой подход ориентирован на стеганосистемы, в которых реализуется скрытая передача априорно неизвестной получателю информации, причем пассивный нару-

шитель пытается в процессе наблюдения за каналом открытой связи обнаружить факт наличия КСС и, в случае установления последнего, стремится раскрыть содержание скрытого сообщения в перехваченном контейнере.

Известно большое количество работ по синтезу стеганосистем, авторы которых предлагают разнообразные способы встраивания данных в избыточные по своей природе контейнеры [21, 24, 63, 64]. При этом количество информации, встроенность которой остается незамеченной, оценивается с помощью дополнительно введенных критериев уровня скрытости (см. предыдущую главу). Существующие на сегодняшний день оценки СПС таких стеганоканалов, однако, не учитывают возможные случайные и умышленные искажения контейнеров при их передаче по каналу связи.

## 4.2. Информационное скрытие при активном противодействии нарушителя

В рамках первого подхода к оценке СПС, рассмотрим общую формулировку задачи информационного скрытия в случае активного противодействия со стороны нарушителя. Основные результаты этого подхода были получены в работе [61] и применены в [5]. Рассмотрим некоторые их них.

### *4.2.L Формулировка задачи информационного скрытия при активном противодействии нарушителя*

Рассмотрим обобщенную структурную схему стеганографической системы передачи скрытых сообщений, представленную на рис. 4.1.

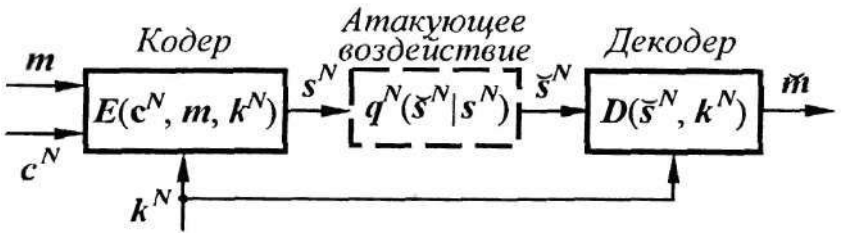



**Рис. 4.1. Обобщенная структурная схема стеганосистемы при активном противодействии нарушителя**

В этой схеме скрываемые сообщения $m$ равномерно распределены во множестве сообщений $M$ и должны быть безошибочно переданы декодеру. Передающая сторона подает пустой контейнер $c^N$ (который представляет собой последовательность с $N$ независимо и идентично распределенных отсчетов в соответствии с распределением контейнера $p(c)$\ секретный ключ $k^N$ (каждый символ $k_i$ которого независимо и равномерно распределен по функции $p(k)$), и сообщения $m$ на вход кодера. Последний формирует стеганограмму $s^N$, которая передается получателю по незащищенному каналу связи.

Стеганограмма $s$ перехватывается и обрабатывается нарушителем с целью разрушения или удаления сообщения $m$. Искаженную нарушителем стеганограмму пометим как $\tilde{s}^N$, а атакующее воздействие — условной функцией распределения

$$q^N\left(\tilde{s}^N \middle| s^N\right).$$

Эта обработка включает, как частный случай, формирование искаженной стеганограммы в виде

$$\tilde{s}^N = q^N\left(s^N\right),$$

где $q^N$ —детерминированное изображение.

Основное предположение: нарушитель знает распределение всех изменений в стеганосистеме и, собственно, описание стеганосистемы, но не знает используемого секретного ключа (принцип Керхгофса для систем защиты информации).

Пусть контейнер *c,* стеганограмма *s* и модифицированная нарушителем стеганограмма *s* принадлежат одному множеству $C$ ($c, s, \tilde{s} \in C$). Декодер получателя вычисляет оценку $m$ первичного скрытого сообщения *m.* Если $m \neq \tilde{m}$, то атакующий сумел разрушить информацию, которая защищалась стеганографической системой.

Формально определим внесенные искажения в стратегиях передающей стороны и нарушителя. Это завершает математическое описание стеганосистемы и позволяет определить скорость безошибочной передачи для схемы, представленной на рис. 4.1.

Определение 4.1

Функция искажения, вносимого отправителем сообщения, представляет собой неотрицательную функцию

$$d_1: C \times S \to \Re_+.$$

**Определение** 4.2

Функция искажения, вносимого атакующей стороной, представляет собой неотрицательную функцию

$$d_2: S \times \tilde{S} \to \Re_+.$$

Функция искажения $d_1$ ограничена:

$$d_{1,\max} = \max_{(c,s)\in C\times S} d_1(c,s) < \infty$$

Кроме того, данная мера искажения симметрична : $d_1(c, s) = d_1(s, c)$ для всех $c, s \in C = S$. Выполнение равенства $d_1(c, s) = 0$ означает совпадение: $c = s$. Если $d_1(c, s) = 1$, то контейнер-результат не соответствует контейнеру-оригиналу.

Функции искажения $d_i$, $i \in \{1; 2\}$ распространяются на искажения символьных последовательностей с длиной блоков $N$:

$$d_i^N(x^N, y^N) = N^{-1} \cdot \sum_{j=1}^{N} d_i(x_j, y_j).$$

Назовем искажение контейнера *c,* вызванное встраиванием в него скрытого сообщения $m$, ***искажением, вызванным кодированием,*** **а искажение, вызванное атакующими действиями нарушителя,—** ***искажением, вызванным атакующим воздействием.***

**Определение 4.3**

Стеганосистема с длиной блока $N$, приводящая к искажению, вызванному кодированием, уровень которого не превышает $A_1$, является совокупностью множеств скрываемых сообщений *M* с количеством элементов (мощностью) $|M|$, контейнеров $C$, стеганограмм $S \sim \tilde{S} \sim C$ и ключей *K,* а также определенных на них функ-

ций кодирования $E$ и декодирования $D$. При этом $E$ — отображение контейнера $c^N$, сообщения $m$ и ключа $k^N$ в стеганограмму:

$$E: C \times M \times K \to C, s^N = E(c^N, m, k^N).$$

Это отображение ограничено величиной среднего искажения $Д\backslash$, вызванного кодированием:

$$\sum_{c^N \in C} \sum_{m \in M} \sum_{k^N \in K} |M|^{-1} \cdot p(c^N, k^N) \cdot d_1^N [c^N, E(c^N, m, k^N)] \le A_1. \qquad (4.1)$$

Отображение $D: C \times K \to \breve{M}$ — декодирующее отображение принятой стеганопоследовательности $\breve{s}^N$ и ключа $k^N$ в декодированное сообщение $\breve{m} = D(\breve{s}^N, k^N)$.

Таким образом, величина $Д\backslash$ характеризует максимально допустимую степень искажения контейнера при встраивании в него скрываемого сообщения. Несмотря на то, что данное определение формально описывает стеганосистемы блочного типа, на практике оно может быть расширено и на стеганосистемы потокового типа, в которых окно обработки описывается скользящим блоком длиной $N$. В этом случае параметр $N$ стеганосистемы может быть назван длиной кодового ограничения стеганосистемы (по аналогии с непрерывными кодами).

В большинстве случаев искажение $A_1$ является малым, поскольку априорно принимается, что результат встраивания в контейнер сообщения должен быть незаметным для постороннего лица (в том числе и нарушителя). В стеганосистемах, в которых контейнер представляет собой полезный для получателя информационный сигнал и качество которого необходимо сохранить, величина $A_1$ ограничивается. В системах ЦВЗ требование минимизации $A_1$ формулируется как требование прозрачности водяного знака, свидетельствующего о принадлежности контейнера [5].

Кроме того, определение ограничения искажения (4.1) содержит усреднение по отношению к распределению $p(c^N, k^N)$ и по отношению к равномерному распределению сообщений. Такой выбор сделан для удобства, поскольку это позволяет использовать классические положения теории Шеннона [60].

Распределение $p(c^N, k^N)$ и выбор отображения $E$ определяют конкретный вид распределения $p(s^N)$ множеств формируемых стеганограмм.

Определение 4.4

Атакующее воздействие (без памяти), приводящее к искажению $A_2$, описывается условной функцией распределения $q^N(\breve{s}^N \mid s^N)$ из множества $S$ к множеству $\breve{S}$, такой, что выполняется условие

$$\sum_{s^N \in S} \sum_{\breve{s}^N \in \breve{S}} d_2^N(s^N, \breve{s}^N) \cdot q^N(\breve{s}^N \mid s^N) \cdot p(s^N) \le A_2. \qquad (4.2)$$

По определению, $A_2$ является максимальной величиной искажения стеганограммы, вызванного умышленными действиями нарушителя. Физический смысл ограничения величины $A_2$ заключается в следующем. В системах ЦВЗ нарушитель, пытаясь удалить водяной знак из заверенного контейнера, вынужден сам уменьшать величину $A_2$, чтобы существенно не исказить ценный для него контейнер. В других стеганосистемах величина $A_2$ ограничивается имеющимся у атакующего энергетическим потенциалом установления помех, возникающими помехами для других каналов связи при использовании общего ресурса и другими причинами.

Логично предположить, что для реальных стеганосистем обычно выполняется соотношение $A_2 \geq A_1$ [5].

В соответствии с определением 4.4, атакующее воздействие описывается и ограничивается усредненными искажениями между множествами $S$ и $\breve{S}$.В других случаях, если атакующий знает описание функции $E$, то атакующее воздействие описывается и ограничивается усредненным искажением между множествами $C$ и $\breve{S}$:

$$\sum_{\substack{c^N, m, \\ k^N, \breve{s}^N}} d^N\left(c^N, \breve{s}^N\right) \cdot q^N\left[\breve{s}^N \middle| E\left(c^N, m, k^N\right)\right] \cdot p\left(c^N, k^N\right) \leq A_2. \qquad (4.3)$$

Определение $A_2$ в соответствии с выражением (4.3) допускает, что нарушителю известны точные вероятностные характеристики контейнеров. Как будет показано дальше, это обстоятельство существенно усложняет задачу обеспечения защищенности скрываемой информации, поэтому в устойчивых стеганосистемах используются разные методы скрытия от нарушителя характеристик используемых контейнеров. Например, такие методы включают использование для встраивания подмножества контейнеров с вероятностными характеристиками, отличающимися от характеристик всего множества известных нарушителю контейнеров, или рандомизированную компрессию сигнала контейнера перед встраиванием в него скрываемого сообщения [15]. Поэтому вычисление искажения $A_2$ в соответствии с определением (4.4) является более универсальным, поскольку нарушитель всегда имеет возможность изучать вероятностные характеристики наблюдаемых стеганограмм.

Имея описание стеганосистемы и атакующего воздействия $q^N(\breve{s}^N \mid s^N)$, можно описать состязание (игру) между передающей и атакующей сторонами.

Определение 4.5

Информационно-скрывающее состязание, которое приводит к искажениям $(A_1, A_2)$, описывается взаимосвязью используемой стеганосистемы, вызывающей искажение кодирования $A_1$, и атакующего воздействия, вызывающего искажение $A_2$. Скорость передачи скрытых сообщений по стеганоканалу определяется в виде $R = N^{-1} \cdot \log |M|$. При этом скорость передачи $R$ выражается через среднее количество бит скрываемого сообщения, которые безошибочно передаются (переносятся) одним символом (пикселем, отсчетом) стеганопоследовательности $s^N$.

Это определение созвучно "классическому" определению скорости передачи обычных сообщений по каналу открытой связи, которая выражается в среднем количестве безошибочно переданных бит за одно использование канала [60, 65, 70].

Вероятность разрушения скрытого сообщения (среднюю вероятность ошибки) в стеганопоследовательности длиной $N$ определяют как

$$P_{\text{br.}}^N = |M|^{-1} \cdot \sum_{m \in M} P\left[D\left(\breve{S}, K\right) \neq m \middle| M = m\right], \qquad (4.4)$$

где скрываемые сообщения $m$ равномерно выбираются из множества $M$. Вероятность $P_{\text{br.}}^N$ является усредненной на множестве всех сообщений вероятностью того, что атакующий успешно исказит скрыто передаваемое сообщение. Атакующий достигает успеха в информационном состязании, если декодированное во время приема сообщение не совпадает со встроенным в контейнер скрываемым сообщением, или же декодер неспособен принять однозначное решение.

Теоретически достижимую скорость безошибочной передачи скрываемых сообщений и скрытую пропускную способность при искажениях не более, чем $(A_1, A_2)$, предлагается определить следующим образом.

Определение 4.6

Скорость $R$ безошибочной передачи скрываемых сообщений является достижимой для искажений не более, чем $(A_1, A_2)$, если существует стеганосистема с длиной блока $N$, которая приводит к искажению кодирования не более $A_1$ на скорости $R_N > R$, такая что $P_{\text{br.}}^N \to 0$ при $N \to \infty$ при любых атаках нарушителя, приводящих к искажениям не более $A_2$

Определение 4.7

Скрытая пропускная способность $B(A_1, A_2)$ является супремумом (верхним пределом) всех достижимых скоростей безошибочной передачи скрываемых сообщений при искажениях не более $(A_1, A_2)$

Таким образом, СПС является *верхним пределом* скорости безошибочной передачи скрываемых данных, при которой искажения контейнера, вызванные встраиванием в него указанных сообщений $(A_1)$ и действиями нарушителя по разрушению этих сообщений $(A_2)$, не превышают заданных величин.

Как и ПС каналов передачи открытых сообщений, ПС каналов передачи скрываемых сообщений определяется в идеализированных условиях, при которых задержка кодирования/декодирования бесконечна (то есть $N \to \infty$), статистика контейнеров, скрываемых сообщений, стеганограмм и ключей точно известна, сложность построения стеганосистемы не ограничена.

Совершенно очевидно, что такая пропускная способность канала скрытой связи имеет смысл теоретического предела, указывающего области, в которых существуют и, соответственно, не существуют стеганосистемы при заданных величинах искажений. Известно, что скорости реальных систем передачи открытых сообщений могут только приближаться к величине ПС открытых каналов, причем по мере приближения к ней вычислительная сложность реализации систем передачи возрастает сначала приблизительно по линейной, а затем — по квадратичной и далее по экспоненциальной зависимости от длины блока кодирования $N$ [60].

Вполне вероятно, что аналогичные зависимости возрастания сложности справедливы и для стеганосистем в меру приближения скорости передачи скрываемых данных к величине СПС. Это предположение подтверждается имеющимся опытом построения стеганосистем [5]. Известно, что попытки увеличить скорость передачи скрываемых данных приводят к существенному усложнению методов скрытия информации [63, 66].

### *4.2.2. Скрывающее преобразование*

Для полного представления стеганосистемы и условий ее функционирования представим формальное описание *скрывающего преобразования,* выполняемого при встраивании информации в контейнер, и *атакующего влияния,* осуществляемого нарушителем для противодействия скрытой передаче. Для этого рассмотрим вспомогательную случайную последовательность и, определенную на множестве $U$.

Физически последовательность $u$ описывает результат преобразования скрываемого сообщения $m$ с целью его адаптации к встраиванию в контейнер заданного

формата. Следует отметить, что в то время как в стеганосистеме ключи и стеганограммы представляют собой последовательности одинаковой длины $N$, длина скрываемых сообщений, их алфавит и вероятностное распределение в преобладающем большинстве случаев не совпадают с соответствующими характеристиками указанных последовательностей.

Определим вспомогательное множество

$$O = \{(c, k) \in C \times K : p(c, k) > 0\}$$

Тогда мощность множеств $U$ должна удовлетворять условию:

$$|U| \le |S| \cdot |O| + 1$$

В общем виде скрывающее преобразование, используемое отправителем для встраивания скрываемого сообщения в контейнер, определяется следующим образом.

**Определение 4.8**

Скрывающее преобразование, вызывающее искажение кодирования $A_1$, описывается условной функцией распределения $\tilde{q}(s,u|c,k)$ отображения из множества $C \times K$ в множество $C \times U$, такой, что выполняется условие

$$\sum_{c,s,k,u} d_1(c,s) \cdot \tilde{q}(s,u|c,k) \cdot p(c,k) \le A_1$$

Расширение скрывающего преобразования без памяти длиной $N$ описывается следующей условной функцией:

$$\tilde{q}^N\left(s^N, u^N \middle| c^N, k^N\right) = \prod_{i=1}^{N} \tilde{q}\left(s_i, u_i \middle| c_i, k_i\right). \qquad (4.6)$$

Для успешного скрытия информации от квалифицированного нарушителя целесообразно использовать не одно, а множество скрывающих преобразований сообщений.

Определение 4.9

Обобщенное скрывающее преобразование, приводящее к искажению кодирования не более величины $A_1$, состоит из множества $\psi$ всех скрывающих преобразований, удовлетворяющих условию (4.5).

Обобщенное скрывающее преобразование описывает все возможные варианты действий отправителя при встраивании сообщений $m$ в контейнер таким образом, чтобы величина искажения кодирования не превышала допустимую $A_1$. Следует отметить, что в стеганографии важно, чтобы у скрывающего информацию существовало множество возможных вариантов, среди которых он равновероятно и непредвиденно для нарушителя выбирает конкретный вариант скрытия сообщения, требующего защиты.

Для анализа стеганосистемы удобно записать функцию $\tilde{q}$ в форме произведения функций распределения:

$$\tilde{q}(s,u|c,k) = p(s|c,u,k) \cdot p(u|c,k),$$

где $p(s|c,u,k)$ относится к "основному" скрывающему преобразованию, а $p(u|c,k)$ — к "вспомогательному" скрывающему преобразованию.

### *4.2.3, Атакующее воздействие*

Рассмотрим формальное описание действий нарушителя относительно преобразования перехваченной стеганограммы $s$ в искаженную стеганограмму $\tilde{s}$ с целью разрушения содержащейся в ней скрытой информации.

**Определение 4.10**

Атакующее воздействие, вызывающее искажение $A_2$, описывается условной функцией распределения $q(\tilde{s}|s)$ отображения из множества $S$ в множество $\tilde{S}$, такой, что выполняется условие

$$\sum_{s,\tilde{s}} d_2(s,\tilde{s}) \cdot q(\tilde{s}|s) \cdot p(s) \le A_2 .$$

Расширение атакующего воздействия без памяти длиной $N$ описывается условной функцией вида

$$q^N\left(\tilde{s}^N \middle| s^N\right) = \prod_{i=1}^{N} q\left(\tilde{s}_i | s_i\right).$$

**Определение 4.11**

Обобщенное атакующее воздействие, вызывающее искажение не более величины $A_2$, состоит из множества ψ всех атакующих воздействий, которые удовлетворяют условию (4.8).

Аналогично набору вариантов действий передающей стороны, у атакующего также есть свой набор атакующих воздействий (множество ψ). Нарушитель, перехватив стеганограмму, пытается выбрать такое атакующее воздействие из множества ψ, которое максимизировало бы вероятность разрушения скрытой в ней информации.

## 4.3. Скрытая пропускная способность при активном противодействии нарушителя

### *4.3.1. Основная теорема информационного скрытия при активном противодействии нарушителя*

Исследуем скрытую пропускную способность (СПС) в случае активного противодействия нарушителя, который стремится разрушить скрыто передаваемую информацию. Информационно-скрывающее состязание между передающей и атакующей сторонами удобно описать методами теории игр [67, 68, 97]. В теории игр обычно идет речь о играх двух сторон (лиц); считается, что игры ограничены, то есть, оба играющих в обоих случаях могут делать только определенное количество шагов, и игра заканчивается после ограниченного количества шагов. Отсюда выте-

кает ограниченность количества стратегий обоих играющих. Под понятием *стратегии* подразумевается такая система правил, с помощью которой задается действие (действия) игрока в определенной ситуации. Целью теории игр является нахождение лучшей стратегии отдельных игроков.

Цена игры в данном случае равна величине СПС, для максимизации которой отправитель информации оптимальным образом формирует скрывающее преобразование. Для минимизации СПС атакующий синтезирует оптимальное атакующее воздействие. Величина СПС может быть получена последовательным соединением скрывающего преобразования и атакующего воздействия. Для того чтобы оценить величину скрытой ПС для стеганосистемы с двоичным алфавитом, приведем исследования теоретико-игровых аспектов проблемы скрытия информации стеганосистемами.

Рассмотрим *основную теорему информационного скрытия при активном противодействии нарушителя* [61]. Для любых стеганосистем произвольной сложности и любых атак без памяти данная теорема ограничивает сверху скорость безошибочной передачи для скрывающего информацию при условии, что атакующий знает описание скрывающего преобразования, а получатель знает описания как скрывающего преобразования, так и атакующего воздействия. Данное условие на самом деле не является трудноосуществимым, как это может показаться на первый взгляд.

Даже если стратегии действий отправителя информации и атакующего неизвестны, но стационарны, то можно утверждать, что как атакующий, так и получатель потенциально способны их определить, обработав достаточно большой объем статистического материала. Это предположение совершенно реалистично, хотя и не всегда может быть достигнуто на практике, учитывая высокую вычислительную сложность [5].

Предварительно рассмотрим два утверждения, которые устанавливают области существования стеганосистем, потенциально способных безошибочно передавать скрываемую информацию при заданном атакующем воздействии [5].

В дальнейшем будем обозначать через *H(x)* энтропию переменной $x$, через $I(x; y)$ — полное количество информации между $x$ и $y$, а через $I(x; y \mid z)$ — условное полное количество информации между $x$ и $y$, обусловленных $z$ [68].

Утверждение 4.1

Зафиксируем атакующее воздействие $q(\tilde{s}|s)$ и изберем скрывающее преобразование $\tilde{q}(s,u|c,k)$, которое максимизирует количество информации вида

$$J(\tilde{q}, q) = I(u; \tilde{s}|k) - I(u; c|k) \qquad (4.10)$$

над $\tilde{\psi}$. Для любого сколь угодно малого значения $\xi > 0$ и достаточно большого значения $N$ существует стеганосистема с длиной блока $N$, которая обеспечивает вероятность разрушения скрываемых сообщений $P_{\text{br}}^{N} < \xi$ для множества скрываемых сообщений мощностью

$$|M| < 2^{N \cdot \left[ I(u;\tilde{s}|k) - I(u;c|k) - \xi \right]}.$$

Утверждение 4.2

Пусть стеганосистема с длиной блока $N$ способна безошибочно передавать скрываемые сообщения со скоростью $R = N^{-1} \cdot \log|M|$ бит/элемент контейнера при ата-

кующем воздействии $q(\tilde{s}|s)$. Если для любого $\xi > 0$ стеганосистема обеспечивает вероятность разрушения скрываемых сообщений $P_{ош}^{N} < \xi$ при $N \to \infty$, то существует конечный алфавит $U$ и такое скрывающее преобразование $\tilde{q}(s,u|c,k)$, что исполняется неравенство

$$R \le I(u;\tilde{s}|k) - I(u;c|k)$$

**Теорема 4.1**

Пусть атакующему известно описание обобщенного скрывающего преобразования $\tilde{\psi}$, а получателю известно описание обобщенного скрывающего преобразования $\tilde{\psi}$ и обобщенного атакующего воздействия $\psi$. Для любого информационно-скрывающего состязания, приводящего к искажению не более, чем $(A_1, A_2)$, скорость передачи $R$ скрываемых сообщений достижима тогда и только тогда, когда $R < \underline{B}$.

Величина $\underline{B}$ определяется как

$$\underline{B} = \max_{\tilde{q}(s,u|c,k)\in\tilde{\psi}} \min_{q(\tilde{s}|s)\in\psi} J(\tilde{q},q), \qquad (4.11)$$

где $u$ — случайная переменная над произвольным конечным алфавитом $U$; переменные $(u, c, k) \to s \to \tilde{s}$ образуют марковскую цепь, представляющую собой частную форму следующей марковской цепи: $u \to (c, s) \to \tilde{s}$, характеристики которой рассматриваются в [69]. Количество информации $J(\tilde{q},q)$ определяется выражением (4.10).

Таким образом, теорема (4.1) определяет величину ***нижнего предела*** скрытой ПС в условиях, когда все участники информационного состязания знают стратегии действий друг друга. Необходимо отметить, что в данной теореме определяется величина СПС стеганоканала, о существовании которого атакующему известно. Данная СПС равна среднему количеству бит информации на один элемент контейнера, которую нарушитель не может разрушить, избирая любую стратегию противодействия из имеющегося множества $\psi$ при искажении контейнера не более величины $A_2$

Доказательство этой теоремы сводится к следующему. Зафиксируем атакующее воздействие $q \in \psi$. В утверждении (4.1) указывается, что все скорости безошибочной передачи скрываемых сообщений, менее $\max_{\tilde{q}\in\tilde{\psi}} J(\tilde{q},q)$, достижимы.

Утверждение (4.2) содержит обратный результат, то есть, достоверная передача выше этой скорости невозможна, поскольку атакующий осведомлен с распределением $\tilde{q}$, которое минимизирует скорость передачи.

Далее показано, что в важном специальном случае, когда $k = c$ (то есть, секретным ключом стеганосистемы является описание используемого контейнера, а сам контейнер известен получателю), нет потери оптимальности при ограничении кодера стеганосистемы видом, представленным на рис. 4.1.

**Следствие**

В случае $k = c$ выбор значения переменной $u$ оптимален тогда и только тогда, когда стеганосистема $s$ может быть записана в форме $s = E(c, u)$, где отображение

$E(c, \bullet)$ обратно для всех значений $c$. В частности, выбор $u = s$ оптимален. Скрытая ПС в этом случае определяется следующим образом:

$$B = \max_{p(s|c)} \min_{q(\breve{s}|s)} I(s;\breve{s}|c) = \min_{q(\breve{s}|s)} \min_{p(s|c)} I(s;\breve{s}|c). \quad (4.12)$$

Это следует из того, что, когда $k = c$, выражение (4.10) может быть записано в виде

$$J(\tilde{q}, q) = I(u;\breve{s} \mid c) = I(u;\breve{s} \mid c) - I(u; c \mid c) = I(u, c;\ \breve{s} \mid c) \le I(s;\breve{s} \mid c). \quad (4.13)$$

Следовательно, вполне логично, что величина скрытой ПС равна взаимной информации между стеганограммой $s$ и искаженной стеганограммой $\breve{s}$ при условии, что отправителю и получателю скрываемой информации известен пустой контейнер $c$.

Для практических систем защиты информации, если секретным ключом стеганосистемы является описание используемого контейнера, возникают две проблемы [5]. Во-первых, получатель должен знать контейнер-оригинал, что ограничивает возможную область применения таких стеганосистем. Во-вторых, отправитель и получатель скрываемых сообщений должны использовать секретную ключевую информацию очень большого объема, что не всегда удобно на практике.

## *4.3.2. Свойства скрытой пропускной способности стеганоканала*

Рассмотрим свойства СПС, описанные в [5]. Скрытая пропускная способность — это функция аргументов $A_1$ и $A_2$, что удобно выразить в виде $B(A_1, A_2)$, и характеризируется следующими свойствами.

- Величина $B(A_1, A_2)$ монотонно возрастает при увеличении уровня искажения, вызванного кодированием $(A_1)$ и монотонно убывает при возрастании искажения, вызванного атакующим воздействием $(A_2)$
- Функция $B(A_1, A_2)$ выпуклая по аргументу $A_2$
- Величина $B(A_1, A_2)$ ограничена сверху энтропией искаженной стеганограммы $\breve{s}$ и энтропией контейнера $c$:

$$B(A_1, A_2) \le \max_{\tilde{q} \in \tilde{\Psi}} \min_{q \in \Psi} H(\breve{s}) \le H(s) \le H(c) \le \log|C|.$$

Данное свойство очевидно, поскольку СПС не может быть больше энтропии искаженной стеганограммы $\breve{s}$. В свою очередь, в силу возможной потери информации из-за атакующего воздействия, величина $H(\breve{s})$ не может быть больше энтропии стеганограммы $s$, а $H(s)$ из-за возможной потери информации при встраивании скрываемых сообщений не может превышать энтропию $H(c)$ пустого контейнера $c$.

Из теории информации известно, что энтропия источника не может превышать логарифм от мощности его алфавита [70]. Поскольку чаще всего используются контейнеры в виде существенным образом избыточных изображений или аудиосигналов, то для таких контейнеров выполняется неравенство $H(c) << \log|C|$, что существенно уменьшает возможное значение СПС. Следовательно, в стеганосистеме чем более близки характеристики дискретных контейнеров к распределению Бернулли (или непрерывных контейнеров к распределению Гаусса), тем большая величина СПС может быть достигнута.

- Величина $B(0, A_2) = 0$ для любых значений атакующего искажения $A_2$, поскольку $A_1 = 0$ означает, что $s = c$, то есть, контейнер-оригинал полностью совпадает со стеганограммой (никакая скрываемая информация не передается).
- Если допустимо достаточно большое атакующее искажение $A_2$, то для любого искажения $A_1$ может быть построена атака нарушителя, в которой стеганопоследовательность $\bar{s}^N$ формируется независимо от $s^N$. Следовательно, в последовательности $\bar{s}^N$ устранены все следы скрытого сообщения, и СПС равна нулю для любых значений искажения кодирования $A_1$. Таким образом, если атакующий имеет возможность подавлять канал передачи скрываемых сообщений неограниченно мощной помехой, то он гарантированно разрушит сообщения, которые были переданы. Однако во многих практических случаях информационного скрытия у нарушителя отсутствует такой энергетический потенциал, или же, в случае его наличия, им невозможно воспользоваться в полной мере.

## *4.3.3. Комментарии подученных результатов*

Рассмотрим выводы из теоремы (4.1) и комментарии свойства скрытой пропускной способности [5].

- Теорема (4.1) определяет, что установление теоретической вероятности скрытой безошибочной передачи информации и теоретической вероятности противодействия этому сводится к вычислению величины СПС $B$ при известных стратегиях сторон и сравнению ее с необходимой скоростью передачи скрываемой информации $R$. Если СПС окажется меньше необходимой скорости, то даже теоретически не существует способа передачи скрываемых сообщений без искажений, и задача атакующего относительно разрушения произвольных стеганосистем гарантированно будет решаться.

  Оптимальная атака нарушителя заключается во внесении такого искажения $A_2$ при котором величина СПС меньше необходимой скорости передачи скрываемых сообщений. Оптимальная стратегия скрывающего информацию сводится к избранию такого кодирования и такой величины вызванного им искажения $A_1$, при которых с учетом искажения $A_2$ необходимая скорость безошибочной передачи не будет превышать СПС. Это означает, что теоретически существует такой способ безошибочной передачи. Однако теоретическая возможность еще не означает, что передающая сторона будет способна реализовать ее на практике. Например, разработчик стеганосистемы может не знать оптимальных принципов ее построения (они еще не открыты), или из-за ограниченности в вычислительных ресурсах он не может себе позволить оптимальную обработку, или требования к своевременности доставки скрываемых сообщений ограничивают длину $N$ блока кодирования и т.п.

  Таким образом, успех любой из состязающихся сторон в окончательном итоге будет определяться соотношением между скоростью передачи $R$ и величинами искажения $A_1$ и $A_2$ контейнера, в котором скрывается информация.

  Рассмотренная теорема информационного скрытия при активном противодействии нарушителя напоминает фундаментальную теорему Шеннона (C. Shannon), в которой определяется, что существует способ безошибочной передачи сообщений по каналу с помехами, если скорость передачи меньше пропускной способности канала, и невозможна достоверная передача со скоростью, превышающей пропускную способность канала. Шеннон также показал, что существуют зависимости между отношением мощности полезного сигнала

к мощности помех в канале связи и величиной скорости безошибочной передачи сообщений по этому каналу. Аналогично этому, в информационно-скрывающем состязании существуют подобные зависимости между отношениями величин искажения кодирования $A_1$ к величине искажения атакующего воздействия $A_2$ и величиной скорости безошибочной передачи скрытых сообщений по стеганоканалу. Но при внешнем сходстве, у задач открытой и скрытой передачи есть существенные отличия. Открытая связь осуществляется в условиях влияния случайных помех канала связи, а передача скрытой информации должна быть обеспечена даже в условиях оптимизированного намеренного противодействия активного нарушителя.

- Рассмотрим связь задачи информационного скрытия с задачей защиты информации от перехвата в прослушиваемом канале. В 1975 г. американский ученый Вайнер (A.D. Wyner) предложил метод защиты информации от чтения нарушителем, что заложило основы *теории кодового зашумления* [5, 71-73]. Отправитель дискретных сообщений выполняет их случайное избыточное кодирование и передает преобразованные сообщения получателю основным каналом связи. Нарушитель наблюдает их в подслушивающем канале, который является отводом от основного канала. Случайное кодирование построено таким образом, что если в подслушивающем канале есть ошибки, то при декодировании они размножаются и надежно искажают защищаемую информацию.

  Метод кодового зашумления предназначен для систем передачи, в которых основной канал безошибочен. Например, основной канал образован на основе волоконно-оптической линии, а нарушитель пытается вести разведку по каналам побочного электромагнитного излучения и наводок, в которых, в силу их природы, существует большое количество помех. Отметим, что нарушитель знает описание системы кодового зашумления, которая не использует секретной ключевой информации (некриптографический способ защиты).

  Подслушивающий канал характеризуется секретной ПС, представляющей собой максимальную скорость безошибочной передачи основным каналом при условии, что неопределенность для перехватчика максимальна (неопределенность защищаемых сообщений равна энтропии этих сообщений). Однако, если подслушивающий канал менее зашумлен по сравнению с основным каналом, то секретная ПС равна нулю.

  В задаче информационного скрытия атакующий способен на большее, чем обыкновенный перехватчик в подслушивающем канале, поскольку он после перехвата защищаемого сообщения умышленно искажает основной канал. Поэтому основной канал передачи не менее зашумлен, чем подслушивающий. Следовательно, в задаче информационного скрытия с активным нарушителем секретная ПС равна нулю.

- Избрание переменной $u$ независимо от контейнера $c$, как это реализовано в системе ЦВЗ [5], в общем случае неоптимально. Анализ выражения (4.10) показывает, что скорости безошибочной передачи в этом случае ограничены сверху величиной $I(u;\ \tilde{s} \mid \mathbf{k})$.

- Пусть выполняется условие $A_2 \geq A_1$. Если атакующему известно описание контейнера $c^N$, то оптимальная атака заключается только в формировании искаженной стеганограммы в виде $\tilde{s}^N = c^N$. В этом случае выходной сигнал после атаки не содержит никаких следов сообщения, и скрытая ПС равна нулю. На практике это может означать следующее. Если нарушителю известен оригинал защищае-

мой от пиратского копирования информации, то никакие стеганосистемы не защитят авторские или имущественные права производителей этой информации.

Рассмотрим потенциально сильную атаку, в которой атакующий стремится сконструировать достаточно близкую к оригиналу оценку контейнера $\tilde{c}^N$. Если атакующий способен синтезировать искаженную стеганограмму $\tilde{s}$ такую, что $H(\tilde{s} \mid c) < \xi$, то СПС будет ограничиваться сверху величиной

$$I(u;\ \tilde{s} \mid k) - I(u; c \mid k) = I(u;\ c,\ \tilde{s} \mid k) - I(u; c \mid \tilde{s}\ , k) - I(u;\ c,\ \tilde{s} \mid k) - \qquad (4.14)$$

$$I(u;\ \tilde{s} \mid c, k) \le I(u;\ \tilde{s} \mid c, k) \le H(\tilde{s} \mid c, k) \le H(\tilde{s} \mid c) < \xi,$$

для любого $u$. Следовательно, величина скрытой ПС стеганоканала $B(A_1, A_2) < \xi$

Таким образом, если нарушитель способен сформировать достаточно точную оценку контейнера (другими словами, выполняется неравенство $H(\tilde{s} \mid c) < \xi$, где величина $\xi$ достаточно мала), то значение СПС ограничено этой малой величиной. На практике это означает, что, имея заполненный контейнер (стеганограмму), нарушитель может попытаться воссоздать из него с некоторой допустимой погрешностью контейнер-оригинал, из которого изъято скрытое сообщение (это особенно актуально в области защиты с помощью ЦВЗ мультимедийной информации).

## 4.4. Двоичная стеганосистема передачи скрываемых сообщений

Определим величину скрытой ПС стеганосистемы, в которой алфавит скрываемых сообщений, ключей и стеганограмм двоичный: $m = c = k = s = \{0; 1\}$ [5, 61]. Пусть контейнер $c$ формируется источником Бернулли с параметром $p = 0{,}5$ (то есть двоичные символы последовательности контейнера равномерны и независимы друг от друга). Функция искажения $d_1 = d_2$ описывается расстоянием Хемминга: $d(x, y) = 0$, если $x = y$ и $d(x, y) = 1$, если $x \neq y$

Описание контейнера является секретным ключом стеганосистемы ($k = c$) и известно получателю. Пусть стеганограммы формируются в виде $s = c \oplus z$, где операция "$\oplus$" — это сложение по модулю 2. Очевидно, что переменная $z$ будет иметь распределение Бернулли и отображать скрываемое сообщение $m$ с искажением $A_1$

Искажение $A_1$ означает, что каждый символ двоичной последовательности $z$ отличается от соответствующего символа двоичной последовательности $m$ с вероятностью $A_1$ . Преобразование сообщения $m$ в последовательность $z$ выполняется передающей стороной с использованием кодера с искажением $A_1$ . Нарушитель обрабатывает стеганограмму наложением на нее двоичной шумовой последовательности $a$, в которой единичный символ порождается с вероятностью $A_2$ . Получатель суммирует искаженную стеганограмму $\tilde{s}$ с двоичной последовательностью $c$ по модулю 2, и из полученной таким образом двоичной последовательности $\tilde{z}$ декодирует принятое скрываемое сообщение $\tilde{m}$

Особенность такой стеганосистемы заключается в том, что скрываемое в ней сообщение при встраивании искажается с вероятностью искажения $A_1$, и это искажение равно искажению кодирования стеганограммы. Описанная стеганограмма изображена на рис. 4.2.

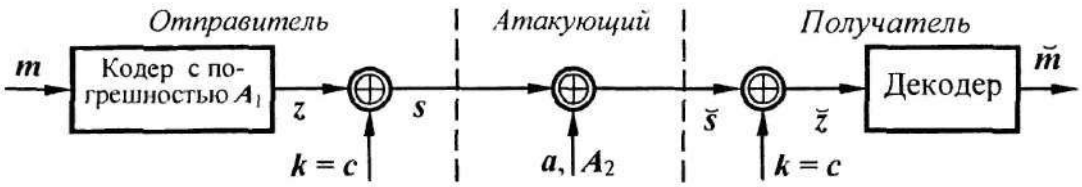



Рис. 4.2. Структурная схема двоичной стеганосистемы

Утверждение 4.3

Для двоичной стеганосистемы при уровнях искажений $A_1, A_2 \le 0{,}5$ скрытая пропускная способность определяется как

$$B = \underline{H}(A_1 * A_2) - \underline{H}(A_2), \qquad (4.15)$$

*где* $\underline{H}(t) = -t \cdot \log(t) - (1-t)\cdot\log(1-t)$; $A_1 * A_2 = A_1\cdot(1-A_2) + A_2\cdot(1-A_1)$.

Для данной стеганосистемы переменную $u$ можно формировать как $u = s$ или $u = z$, причем оба варианта могут быть оптимальными, поскольку в качестве операции встраивания используется операция суммирования по модулю 2 [5].

Оптимальная атака нарушителя определяется в виде $\breve{s} = s \oplus a$, где $a$ — случайная двоичная последовательность, распределенная по закону Бернулли с вероятностью появления единичного символа $-A_2$

При уровнях искажений $A_1 \ge 0{,}5$ и $A_2 < 0{,}5$ СПС равна $B = 1 - \underline{H}(A_2)$. Если $A_2 \ge 0{,}5$, СПС равна нулю.

Необходимо отметить, что при $A_1 = 0{,}5$ СПС не равна нулю независимо от значения $A_2 < 0{,}5$. Это объясняется тем, что при преобразовании скрываемого сообщения $m$ в последовательность $z$ искажение не равновероятно: лицо, которое скрывает информацию, может избрать такое распределение ошибок $A_1$, при котором будет минимизироваться переменная сообщения $m$. Для $A_2 = 0{,}5$ СПС будет равна нулю при любых значениях $A_1$

Несложно заметить, что в этом случае выход $\breve{s}$ канала не зависит от его входа $s$, что означает разрыв канала связи. А если при обрыве канала связи невозможна передача по открытому каналу связи, то тем более невозможна и передача по скрытому каналу, образованному на основе открытого.

Применим следствие теоремы (4.1) для анализа двоичной стеганосистемы. Пусть $z = c \oplus s$, $a = s \oplus \breve{s}$ . Платежная функция имеет вид $I(s;\ \breve{s} \mid \mathbf{c})$. Допустим, что $A_1,\ A_2 \le 0{,}5$.

Шаг 1. Зафиксируем $q(\breve{s} \mid \mathbf{s})$. Для всех $q \in \psi$, получим

$$I(s;\ \breve{s} \mid c) \overset{(a)}{=} H(\breve{s} \mid c) - H(\breve{s} \mid \mathbf{s}, c) \overset{(b)}{=}$$

$$\overset{(b)}{=} H(\breve{s} \mid c) - H(\breve{s} \mid s) = H(\breve{s} \oplus c \mid c) - H(a) = H(z \oplus a \mid c) - \underline{H}(A_2) \overset{(c)}{\le}$$

$$\overset{(c)}{\le} H(z \oplus a) - \underline{H}(A_2) \overset{(d)}{\le} \underline{H}(A_1 * A_2) - \underline{H}(A_2),$$

*где* равенство (a) справедливо относительно определения условного взаимодействия информации; (b) выполняется благодаря тому, что $c \to s \to \breve{s}$ — марковская цепь; неравенство (c) справедливо, поскольку условие уменьшает энтропию. Равенство в (c) достигается тогда и только тогда, когда $z \oplus a$, следовательно, $z$ независима от $c$. Неравенство (d) справедливо, поскольку $z$ и $a$ независимы (в силу того, что

$z \to s \to a$ формирует марковскую цепь и $P[z = 1] \le A_1$). Равенство (d) достигается, если переменная $z$ имеет распределение Бернулли с дисперсией $A_1$. Распределение $p(s \mid c)$ удовлетворяет обоим нестрогим неравенствам и поэтому максимизирует значение $I(s;\ \tilde{s} \mid c)$

Шаг 2. Зафиксируем $p(s \mid c)$. Будем минимизировать $I(s;\ \tilde{s} \mid c)$ над $q(\tilde{s} \mid s)$. При определенном ранее распределении $p(s \mid c)$, $z$ и $s$ независимы. Поскольку $z \to s \to a$ формирует марковскую цепь, $z$ и $a$ также независимы. Имеем

$$I(s;\ \tilde{s} \mid c) = I(s \oplus c;\ \tilde{s} \oplus c \mid c) = I(z;\ z \oplus a \mid c) = H(z) - H(z \mid z \oplus a, c) \overset{(e)}{\ge}$$

$$\overset{(e)}{\ge} H(z) - H(z \mid z \oplus a) = I(z;\ z \oplus a) \overset{(f)}{\ge} \underline{H}(A_1 * A_2) - \underline{H}(A_2),$$

где неравенство (e) справедливо, поскольку условие уменьшает энтропию; неравенство (f) справедливо, потому что $z$ и $a$ независимы и $P[a = 1] \le A_2$ (что становится равенством, когда $a$ — переменная с распределением Бернулли с вероятностью единичного символа $A_2$).

Рассмотренная двоичная стеганосистема похожа на систему шифрования однократной подстановки (шифр гаммирования с бесконечной равновероятностной независимой шифрующей гаммой). При независимой и равномерной последовательности $c$ выполняется равенство $H(z) = H(z \mid s)$, означающее, что данная система удовлетворяет требованию относительно идеальных криптосистем [60]. Следовательно, перехват и анализ криптограммы $s$ не дает атакующему никакой информации относительно защищаемого сообщения $s$.

Кроме этого, указанная двоичная система удовлетворяет также требованию относительно идеальных стеганосистем: распределения $p(c)$ и $p(s)$ идентичны, поэтому нарушитель не в состоянии определить, принадлежат ли перехваченные им данные к распределению $p(c^N)$ пустых контейнеров или же к распределению $p(s^N)$ стеганограммы со встроенным сообщением [15]. Однако при этом отмечается, что в рассмотренной стеганосистеме предусматривается описание контейнеров и, соответственно, стеганограмм распределением Бернулли, а это зачастую нехарактерно для реальных систем скрытия информации [5].

Рассмотрим пример двоичной стеганосистемы с выбором $u = z$. Пусть существует необходимость в скрытой передаче сообщения /и, которое представляет собой оцифрованный речевой сигнал с количеством уровней квантования 8. В общем виде скрываемое сообщение может быть представлено в виде $m = \{m_1, m_2, m_3, m_4, \ldots\}$

Пусть первые несколько отсчетов сообщения в моменты времени дискретизации $t_1, t_2, t_3, t_4, \ldots$ принимают десятичные значения $m_1 = 0$, $m_2 = 8$, $m_3 = 19$, $m_4 = 80$ (рис. 4.3, *а)*.

В двоичной форме скрываемое сообщение запишем как

$$m_1 = 0000\ 0000_2,\quad m_2 = 0000\ 1000_2,\quad m_3 = 0001\ 0011_2,\quad m_4 = 0101\ 0000_2, \ldots$$

Преобразуем двоичную последовательность $m$ в двоичную последовательность $z$ с погрешностью $A_1$. В двоичной системе погрешность кодирования $A_1$ вычисляется по метрике Хемминга. Пусть $A_1 = 1/8$. Следовательно, для формирования последовательности $z = \{z_1, z_2, z_3, z_4, \ldots\}$ лицо, скрывающее информацию, искажает восьмую часть битов последовательности $m$. Для уменьшения искажения скрываемого сообщения ему целесообразно искажать только младшие биты двоичной последовательности $m$.

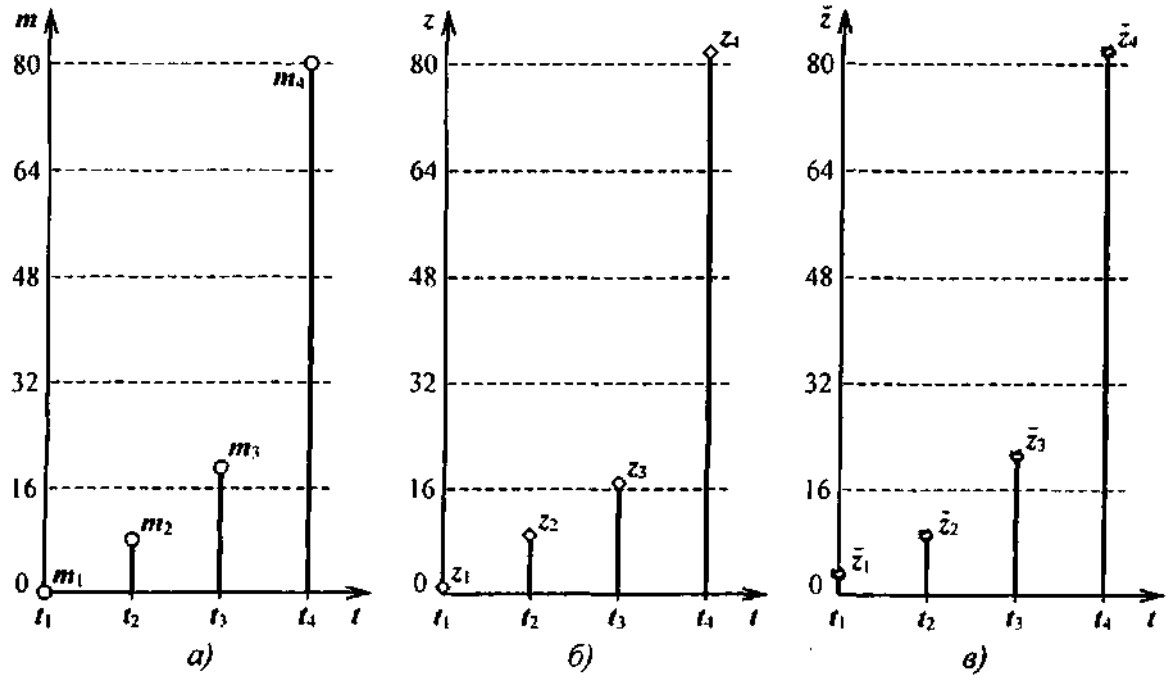



**Рис. 4.3. Пример двоичной стеганосистемы с искажениями $A_1 = 1/8$ и $A_2 = 1/16$**

Пусть скрывающий информацию избрал последовательность $\zeta$ следующего вида:

$$z_1 = 0000\ 0001_2,\quad z_2 = 0000\ 1001_2,\quad z_3 = 0001\ 0001_2,\quad z_4 = 0101\ 0010_2, \ldots .$$

В десятичном виде последовательность $\underline{z}$ изображена на рис. 4.3, *б*.

Допустим, что с помощью генератора ПСЧ был сформирован секретный ключ $k = \{k_1, k_2, k_3, k_4, \ldots\}$

$$k_1 = 0100\ 1101_2,\quad k_2 = 0111\ 0010_2,\quad k_3 = 0101\ 0101_2,\quad k_4 = 0101\ 1001_2, \ldots .$$

Отправитель по правилу $s = k \oplus z$ формирует стеганосистему

$$s_1 = 0100\ 1100_2,\quad s_2 = 0111\ 1011_2,\quad s_3 = 0100\ 0100_2,\quad s_4 = 0000\ 1011_2, \ldots .$$

Пусть искажение $A_2 = 1/16$. Нарушитель случайно формирует двоичную последовательность $a = \{a_1, a_2, a_3, a_4, \ldots\}$, в которой вероятность появления единичных символов составляет $A_2$. Например:

$$a_1 = 0000\ 0010_2,\quad a_2 = 0000\ 0000_2,\quad a_3 = 0000\ 0100_2,\quad a_4 = 0000\ 0000_2, \ldots .$$

Атакующее воздействие — это сложение по модулю 2 стеганограммы $s$ и шумовой последовательности $a$. Следовательно, искаженная стеганограмма $\breve{s} = \{\breve{s}_1, \breve{s}_2, \breve{s}_3, \breve{s}_4, \ldots\}$ будет иметь следующий вид:

$$\breve{s}_1 = 0100\ 1110_2,\quad \breve{s}_2 = 0111\ 1011_2,\quad \breve{s}_3 = 0100\ 0000_2,\quad \breve{s}_4 = 0000\ 1011_2, \ldots .$$

Получатель для формирования принятого сообщения $\breve{z}$ прибавляет по модулю 2 последовательность $k$ и последовательность вида $\breve{z}$:

$$\breve{z}_1 = 0000\ 0011_2,\quad \breve{z}_2 = 0000\ 1001_2,\quad \breve{z}_3 = 0001\ 0101_2,\quad \breve{z}_4 = 0101\ 0010_2, \ldots .$$

В декодере получатель из данной последовательности восстанавливает сообщение $m$. В простейшем случае $\breve{m} = \breve{z}$. Вид последовательности $\breve{z}$ изображен на рис. 4.3, *в*.

Если скрываемое сообщение $m$ представляет собой речевой сигнал, то при указанных величинах искажений $A_1$ и $A_2$ степень приближенности $\tilde{m}$ к $m$ (то есть, качество обеспечиваемой скрытой телефонной связи), для ряда телекоммуникационных задач может быть оценена как удовлетворительная.

## 4.5. Выводы

В данной главе введено одно из ключевых понятий теории передачи информации и, в частности, стеганографических систем (как каналов скрытого обмена данными) — пропускная способность канала передачи скрываемых данных. В процессе обработки зарубежных и отечественных литературных источников выделено два основных подхода к оценке ПС стеганосистем.

- Подход, ориентированный на стеганосистемы, в которых скрытые сообщения должны быть безошибочно переданы адресату в условиях активного противодействия нарушителя. При этом учитывается, что кроме искажений контейнера при встраивании в него конфиденциальных данных, вероятны намеренные его искажения со стороны активного нарушителя и/или искажения, вызванные случайными помехами в канале связи.
- Подход, ориентированный на стеганосистемы, в которых реализуется скрытая передача априорно неизвестной получателю информации, причем пассивный нарушитель пытается в процессе наблюдения обнаружить факт наличия канала скрытой связи и, в случае успеха, стремится раскрыть содержание скрытых данных.

В рамках первого подхода представлены основные задачи информационного скрытия в случае активного противодействия нарушителя; описано скрывающее преобразование, выполняемое при встраивании информации в контейнер, и атакующее воздействие, совершаемое нарушителем для противодействия скрытой передаче; рассмотрена основная теорема информационного скрытия при активном противодействии нарушителя; приведено определение величины скрытой ПС двоичной стеганосистемы.

На основании этого рассмотрены основные свойства скрытой пропускной способности стеганоканала, приведены комментарии о полученных результатах, что позволило заложить обоснованную теоретическую базу для разработки систем стеганографического скрытия конфиденциальной информации.

# Глава 5

# Стеганографические методы скрытия данных и их реализация в системе MathCAD

## 5.1. Вступительные положения

В данной главе рассматриваются стеганографические методы скрытия данных для разных типов информационной среды в качестве стеганоконтейнеров. При этом значительное внимание уделено проблеме практической реализации рассматриваемых методов с использованием современных средств вычислительной техники и программного обеспечения.

В соответствии с [3], во время рассмотрения методов будем обозначать буквой *C* контейнер, представляющий собой последовательность элементов $c_i$ длиной $l_C$. В случае использования в качестве контейнера файла цифрового звука, это будет количество отсчетов за единицу времени, для файла цифрового изображения — последовательность, полученная путем векторизации изображения (то есть, путем развертывания массива всех пикселей изображения в вектор)[1].

Для двоичных массивов контейнеров значения с, могут принимать значения **"0"** или **"1"**; для квантованного изображения или звука (если количество бит, которым кодируется один отсчет, равно 8) — изменяться в диапазоне от 0 до 255 ($2^8 = 256$ градаций); для звука, отсчеты которого кодируются 16-ю битами, — в диапазоне от -32768 до 32767 ($2^{16} = 65536$ градаций); для текстов $c_i$ — это символ кодовой таблицы, код которого может принимать значения от 0 до 255.

Аналогично, будем обозначать буквой *S* заполненный контейнер (стеганограмму) — последовательность элементов $s_j$ длиной $l_S$, а буквой *M*—сообщения длиной $l_M$. которое подлежит скрытию. Если не будет особо оговорено иное, будем считать, что $m_n \in \{0; 1\}$

## 5.2. Классификация методов скрытия данных

Подавляющее большинство методов компьютерной стеганографии (КС) базируется на двух ключевых принципах [3]:

- файлы, которые не требуют абсолютной точности (например, файлы с изображением, звуковой информацией и т.д.), могут быть видоизменены (конечно, до определенной степени) без потери своей функциональности;

[1] В дальнейшем будет показана возможность проведения векторизации и цифрового звука, если последний имеет два и больше каналов.

- органы чувств человека неспособны надежно различать незначительные изменения в модифицированных таким образом файлах и/или отсутствует специальный инструментарий, который был бы способен выполнять данную задачу.

В основе базовых подходов к реализации методов КС в рамках той или иной информационной среды лежит выделение малозначительных фрагментов этой среды и замена существующей в них информации информацией, которую необходимо скрыть. Поскольку в КС рассматриваются среды, поддерживаемые средствами вычислительной техники и компьютерных сетей, то вся информационная среда в результате может быть представлена в цифровом виде [3].

Таким образом, незначительные для кадра информационной среды фрагменты относительно того или иного алгоритма или методики заменяются (замещаются) фрагментами скрываемой информации. Под кадром информационной среды в данном случае подразумевается определенная его часть, выделенная по характерным признакам. Такими признаками зачастую являются семантические характеристики выделяемой части информационной среды. Например, кадром может быть избрано какое-нибудь отдельное изображение, звуковой файл, Web-страница и т.д.

Для существующих методов компьютерной стеганографии вводят следующую классификацию (рис. 5.1) [3, 5].

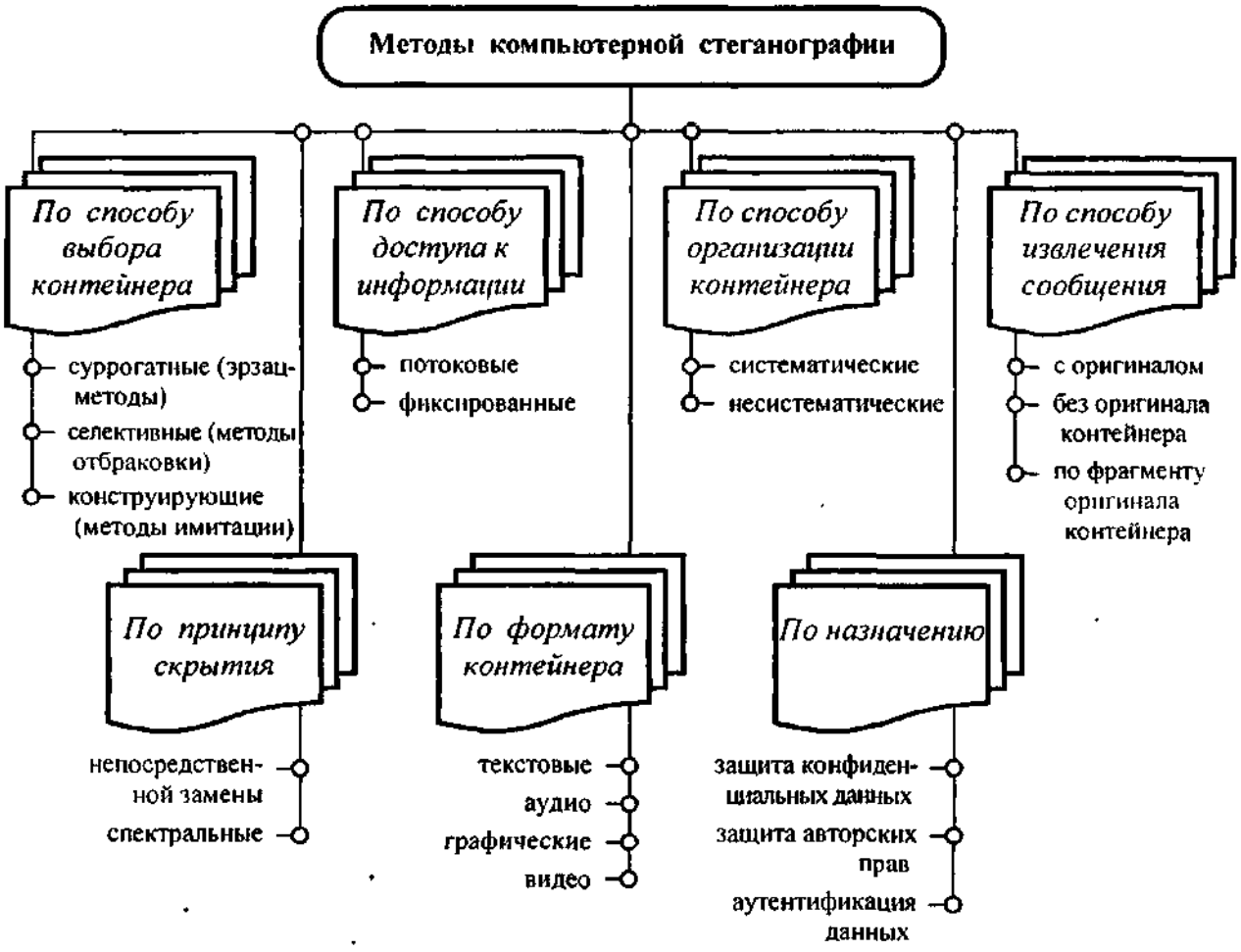



Рис. 5.1. Классификация методов компьютерной стеганографии

Как уже отмечалось в главе 2, **по способу выбора контейнера** различают суррогатные (или так называемые эрзац-методы), селективные и конструирующие методы стеганографии [3].

В *суррогатных {безальтернативных)* методах стеганографии полностью отсутствует возможность выбора контейнера, и для скрытия сообщения избирается первый попавшийся контейнер, — эрзац-контейнер, — который в большинстве случаев не оптимален для скрытия сообщения заданного формата.

В *селективных* методах КС предусматривается, что скрытое сообщение должно воспроизводить специальные статистические характеристики шума контейнера. Для этого генерируют большое количество альтернативных контейнеров с последующим выбором (путем отбраковки) наиболее оптимального из них для конкретного сообщения. Особым случаем такого подхода является вычисление некоторой хэш-функции для каждого контейнера. При этом для скрытия сообщения избирается тот контейнер, хэш-функция которого совпадает со значением хэш-функции сообщения (то есть стеганограммой является избранный контейнер).

В *конструирующих* методах стеганографии контейнер генерируется самой стеганосистемой. При этом существует несколько вариантов реализации. Так, например, шум контейнера может имитироваться скрытым сообщением. Это реализуется с помощью процедур, которые не только кодируют скрываемое сообщение под шум, но и сохраняют модель изначального шума. В предельном случае по модели шума может строиться целое сообщение. Примером может служить метод, реализованный в программе MandelSteg [74], которая в качестве контейнера генерирует фрактал Мандельброта (Mandelbrot fractal), или же аппарат функции имитации [67].

**По способу доступа к скрываемой информации** различают методы для *потоковых {беспрерывных)* контейнеров и методы для ***фиксированных {ограниченной длины)*** контейнеров (более подробно см. раздел 2.3).

**По способу организации** контейнеры, подобно помехоустойчивым кодам, могут быть *систематическими* и *несистематическими* [3]. В первых можно указать конкретные места стеганограммы, *где* находятся информационные биты собственно контейнера, а где — шумовые биты, предназначенные для скрытия информации (как, например, в широко распространенном методе наименее значащего бита). В случае несистематической организации контейнера такое разделение невозможно. В этом случае для выделения скрытой информации необходимо обрабатывать содержимое всей стеганограммы.

**По используемому принципу скрытия** методы компьютерной стеганографии делятся на два основных класса: методы *непосредственной замены* и *спектральные* методы. Если первые, используя избыток информационной среды в пространственной (для изображения) или временной (для звука) области, заключаются в замене малозначительной части контейнера битами секретного сообщения, то другие для скрытия данных используют спектральные представления элементов среды, в которую встраиваются скрываемые данные (например, в разные коэффициенты массивов дискретно-косинусных преобразований, преобразований Фурье, Карунена-Лоева, Адамара, Хаара и т.д.) [5].

Основным направлением компьютерной стеганографии является использование свойств именно избыточности контейнера-оригинала, но при этом следует принимать во внимание то, что в результате скрытия информации происходит искажение некоторых статистических свойств контейнера или же нарушение его структуры. Это необходимо учитывать для уменьшения демаскирующих признаков.

В особую группу можно также выделить методы, которые **используют специальные свойства форматов представления файлов** [3]:

- зарезервированные для расширения поля файлов, которые зачастую заполняются нулями и не учитываются программой;

- специальное форматирование данных (сдвиг слов, предложений, абзацев или выбор определенных позиций символов);
- использование незадействованных участков на магнитных и оптических носителях:
- удаление файловых заголовков-идентификаторов и т.д.

В основном, для таких методов характерны низкая степень скрытости, низкая пропускная способность и слабая производительность.

**По назначению** различают стеганометоды собственно для скрытой передачи (или скрытого хранения) данных и методы для скрытия данных в цифровых объектах с целью защиты авторских прав на них.

**По типам контейнера** выделяют стеганографические методы с контейнерами в виде текста, аудиофайла, изображения и видео.

Рассмотрим подробнее стеганографические методы скрытия данных в неподвижных изображениях, в аудиосигналах и в текстовых файлах.

## 5.3. Скрытие данных в неподвижных изображениях

Большинство исследований посвящено использованию в качестве стеганоконтейнеров именно изображений. Это обусловлено следующими причинами:

- существованием практической необходимости защиты цифровых фотографий, изображений, видео от противозаконного тиражирования и распространения;
- относительно большим объемом цифрового представления изображений, что позволяет встраивать ЦВЗ значительного объема или же повышать устойчивость этого встраивания;
- заранее известным (фиксированным) размером контейнера, отсутствием ограничений, которые накладываются требованиями скрытия в реальном времени;
- наличием в большинстве реальных изображений текстурных областей, имеющих шумовую структуру и наилучшим образом подходящих для встраивания информации;
- слабой чувствительностью человеческого глаза к незначительным изменениям цветов изображения, его яркости, контрастности, содержания в нем шума, искажений вблизи контуров;
- наконец, хорошо разработанными в последнее время методами цифровой обработки изображений.

Однако, как указывается в [5], последняя причина вызывает и значительные трудности в обеспечении стойкости ЦВЗ: чем более совершенными становятся методы компрессии, тем меньше остается возможностей для встраивания посторонней информации.

Развитие теории и практики алгоритмов компрессии изображений привело к изменению представлений о технике встраивания ЦВЗ. Если сначала предлагалось встраивать информацию в незначащие биты для уменьшения визуальной заметности, то современный подход, наоборот, заключается во встраивании ЦВЗ в наиболее существенные области изображений, разрушение которых будет приводить к полной деградации самого изображения. Поэтому абсолютно понятна необходимость учета стеганоалгоритмами не только алгоритмов компрессии изображений, но и свойств зрительной системы человека (ЗСЧ).

### *5.3,1. Основные свойства ЗСЧ, которые необходимо учитывать при построении стеганоалгоритмов*

Свойства ЗСЧ можно разделить на две группы: низкоуровневые ("физиологические") и высокоуровневые ("психофизиологические") [75, 76]. Почти до середины 1990-х г.г. исследователи принимали во внимание, главным образом, низкоуровневые свойства зрения. В последние годы обозначилась тенденция построения стеганоалгоритмов с учетом и высокоуровневых характеристик ЗСЧ.

Выделяют три важнейших *низкоуровневых свойства,* влияющих на заметиость постороннего шума в изображении:

- чувствительность к изменению яркости (контрастности) изображения;
- частотная чувствительность;
- эффект маскировки.

На рис. 5.2 изображена зависимость минимального контраста $\Delta I/I$ от яркости $I$.

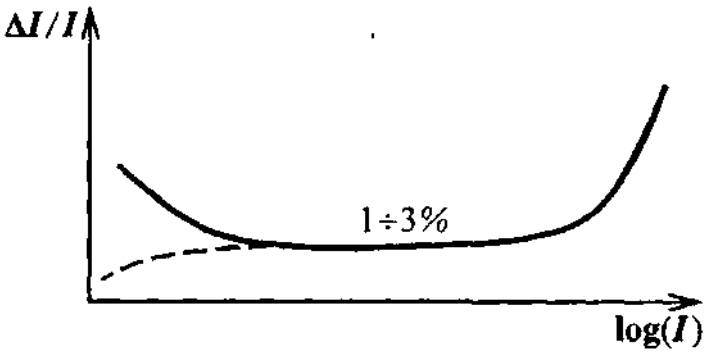



Рис. 5.2. Чувствительность к изменению контраста и порог неразличимости $\Delta I$

Как видно, для среднего диапазона изменения яркости, контраст приблизительно постоянен, тогда как для малых и больших яркостей значение порога неразличимости $(\Delta I)$ возрастает. Установлено, что $\Delta I \approx (0.01 \div 0.03) \cdot I$ для средних значений яркости.

Кроме того, в [5] отмечено, что результаты новейших исследований противоречат "классической" теории и показывают, что при малых значениях яркости порог неразличимости уменьшается, то есть ЗСЧ более чувствительна к шуму в этом диапазоне.

Частотная чувствительность ЗСЧ проявляется в том, что человек намного более восприимчив к низкочастотному (НЧ), чем к высокочастотному (ВЧ) шуму. Это связано с неравномерностью амплитудно-частотной характеристики ЗСЧ.

Элементы ЗСЧ разделяют поступающий видеосигнал, на отдельные составляющие, каждая из которых возбуждает нервные окончания глаза через ряд подканалов. Выделяемые глазом составляющие имеют разные пространственные и частотные характеристики, а также различную пространственную ориентацию (горизонтальную, вертикальную, диагональную) [77].

В случае одновременного влияния на глаз двух составляющих с похожими характеристиками возбуждаются одни и те же подканалы. Это приводит к *эффекту маскировки,* который заключается в увеличении порога обнаружения зрительного сигнала в присутствии другого сигнала, имеющего аналогичные характеристики. Поэтому, аддитивный шум намного заметней на НЧ (однотонных) участках изображения по сравнению с ВЧ участками, то есть, в последнем случае наблюдается маскировка. Наиболее сильно данный эффект проявляется, когда оба сигнала имеют одинаковую ориентацию и место расположения [5].

Частотная чувствительность тесно связана с яркостью. Известно также и выражение для определения порога маскировки на основе известной яркостной чувствительности, что позволяет найти метрику искажения изображения, которая учитывала бы свойства ЗСЧ. Математические модели такого типа хорошо разработаны для случая квантования коэффициентов дискретного косинусного преобразования, поскольку именно оно применяется в стандарте JPEG.

Эффект маскировки в пространственной области может быть объяснен путем построения стохастических моделей изображения. При этом изображение представляется в виде марковского случайного поля, распределение вероятностей которого описывается, например, обобщенным законом Гаусса.

В [5] предлагается следующая обобщенная схема встраивания данных в изображение:

1. Выполняется фильтрация изображения с помощью ориентированных полосовых фильтров. При этом получается распределение энергии по частотно-пространственным компонентам.
2. Рассчитывается порог маскировки на основе знания локальной величины энергии.
3. Масштабируется значение энергии внедряемой информации в каждом компоненте таким образом, чтобы оно было меньше порога маскировки.

Эта схема так или иначе используется многими алгоритмами встраивания данных.

*Высокоуровневые свойства **ЗСЧ*** пока еще редко учитываются при построении стеганоалгоритмов [5]. Они отличаются от низкоуровневых тем, что проявляются "вторично" — обработав первичную информацию от ЗСЧ, мозг выдает команды на "подстройку" зрительной системы под изображение.

Перечислим основные из этих свойств:

- **чувствительность к контрасту** — высококонтрастные участки изображения и перепады яркости обращают на себя больше внимания;
- **чувствительность к размеру** — большие участки изображения более "заметны" по сравнению с меньшими по размеру, причем существует порог насыщенности, когда дальнейшее увеличение размера не играет роли;
- **чувствительность к форме** — длинные и тонкие объекты вызывают больше внимания, чем закругленные и однородные;
- **чувствительность к цветам** — некоторые цвета (например, красный) более "заметны", чем другие; этот эффект усиливается, если фон заднего плана отличается от цветов фигур на нем;
- **чувствительность к месту размещения** — человек склонен в первую очередь рассматривать центр изображения; также внимательней рассматриваются фигуры переднего плана, чем заднего;
- **чувствительность к внешним раздражителям** — движение глаз наблюдателей зависит от конкретной обстановки, от полученных ими перед просмотром или во время его инструкций, дополнительной информации.

В последнее время создано достаточное количество методов скрытия данных в цифровых изображениях, что позволяет провести их классификацию и выделить следующие обобщенные группы [3]:

- методы замены в пространственной области;
- методы скрытия в частотной области изображения;

- широкополосные методы;
- статистические (стохастические) методы;
- методы искажения;
- структурные методы.

Далее рассматриваются особенности, характерные для каждой из выделенных групп. Параллельно приводятся программные модули в системе MathCAD, позволяющие реализовать тот или иной метод, а также промежуточные и конечные результаты соответствующих стеганографических преобразований.

Для каждого модуля даны краткие объяснения относительно его функционирования и использованных функций систем MathCAD. При этом наиболее исчерпывающие объяснения прилагаются к первым рассмотренным методам. В дальнейшем значение функций, содержание которых было раскрыто ранее, не объясняется.

Полное представление о возможностях и правилах использования (синтаксисе) типичных объектов языка MathCAD можно получить, например, из [25, 26]. Кроме того, в данной книге в приложениях A, B, C и D представлена краткая информация относительно встроенных операторов, функций и директив, а также системных переменных и программных операторов системы MathCAD.

## *5.3.2. Скрытие данных в пространственной области*

Алгоритмы, описанные в данном подразделе, встраивают скрываемые данные в области первичного изображения. Их преимущество заключается в том, что для встраивания нет необходимости выполнять вычислительно сложные и длительные преобразования изображений.

Цветное изображение *C* будем представлять через дискретную функцию, которая определяет вектор цвета *c(x, y) для* каждого пикселя изображения $(x, y)$, *где* значение цвета задает трехкомпонентный вектор в цветовом пространстве. Наиболее распространенный способ передачи цвета— это модель RGB, в которой основные цвета — красный, зеленый и синий, а любой другой цвет может быть представлен в виде взвешенной сумы основных цветов.

Вектор цвета $c(x, y)$ в RGB-пространстве представляет интенсивность основных цветов. Сообщения встраиваются за счет манипуляций цветовыми составляющими $\{R(x, y), G(x, y), B(x, y)\}$ или непосредственно яркостью $\lambda(x, y) \in \{0, 1, 2, ..., L_C\}$

Общий принцип этих методов заключается в замене избыточной, малозначимой части изображения битами секретного сообщения. Для извлечения сообщения необходимо знать алгоритм, по которому размещалась по контейнеру скрытая информация.

### 5.3.2.1. Метод замены наименее значащего бита

Метод ***замены наименее значащего бита*** (НЗБ, LSB — Least Significant Bit) наиболее распространен среди методов замены в пространственной области [3, 5, 9, 14, 19,20].

Младший значащий бит изображения несет в себе меньше всего информации. Известно, что человек в большинстве случаев не способен заметить изменений в этом бите. Фактически, НЗБ — это шум, поэтому его можно использовать для встраивания информации путем замены менее значащих битов пикселей изображения битами секретного сообщения. При этом, для изображения в градациях серого (каждый пиксель изображения кодируется одним байтом) объем встроенных данных может составлять 1/8 от общего объема контейнера. Например, в изображение

размером 512×512 можно встроить ~32 кБайт информации. Если же модифицировать два младших бита (что также практически незаметно), то данную пропускную способность можно увеличить еще вдвое.

Популярность данного метода обусловлена его простотой и тем, что он позволяет скрывать в относительно небольших файлах достаточно большие объемы информации (пропускная способность создаваемого скрытого канала связи составляет при этом от 12,5 до 30%). Метод зачастую работает с растровыми изображениями, представленными в формате без компрессии (например, GIF и BMP ) [3][2].

Метод НЗБ имеет низкую стеганографическую стойкость к атакам пассивного и активного нарушителей. Основной его недостаток— высокая чувствительность к малейшим искажениям контейнера. Для ослабления этой чувствительности часто дополнительно применяют помехоустойчивое кодирование.

Перед импортом изображения-контейнера в документ MathCAD его необходимо подготовить в соответствующем редакторе и записать в виде файла в текущий (для формируемого документа MathCAD) каталог (следует отметить, что во избежание возможных проблем с поддержкой кириллицы желательно, чтобы адрес размещения файла на диске, как, собственно, и имя файла, состояли из латинских символов). MathCAD поддерживает форматы BMP, JPEG, GIF, PCX и TGA. Как было указано выше, форматы BMP и GIF, позволяют сохранять изображения практически без потери их качества и потому более пригодны в роли носителей информации.

Рассмотрим структуру BMP-файла: он содержит точечное (растровое) изображение и состоит из трех основных разделов: заголовка файла, заголовка растра и растровых данных.

Заголовок файла содержит информацию о файле (его тип, объем и т.п.). В заголовок растра вынесена информация о ширине и высоте изображения, количество битов на пиксель, размер растра, глубина цвета, коэффициент компрессии и т.д.

Нас в первую очередь будут интересовать *растровые данные* — информация о цвете каждого пикселя изображения. Цвет пикселя определяется объединением трех основных цветовых составляющих: красной, зеленой и синей (сокращенно, RGB). Каждой из них соответствует свое значение интенсивности, которое может изменяться от 0 до 255. Следовательно, за каждый из цветовых каналов отвечает 8 битов (1 байт), а глубина цвета изображения в целом — 24 бита (3 байта).

**Шаг 1**

Импорт графического файла выполняется операцией **Picture** из позиции **Insert** главного меню программы. В модуле, который при этом появился, необходимо заполнить шаблон данных в левом нижнем углу, для чего в двойных кавычках следует ввести имя файла (или же, при необходимости, — полный путь его размещения на диске) и нажать клавишу <Enter>.

Пример цветного изображения, восстановленного с помощью операции **Picture** представлен на рис. 5.3. Это изображение имеет размер **128x128** пикселей, глубина цвета — 24 бита. Для возможности обработки изображения необходимо перевести цветовые характеристики каждого его пикселя в числовую матрицу. Для выполнения этой операции


**Рис. 5.3. Изображение-контейнер**

[2] Вообще, форматы BMP и GIF используют алгоритмы компрессии, но эти алгоритмы простейшие, что позволяет сохранять изображение практически без потери его качества

применяется функция **READRGB**("имя_файла"), возвращающая массив из трех подмассивов, которые, в свою очередь, несут информацию о разложении цветного изображения на цветовые компоненты R, G и B:

**C** := **READRGB**("C.bmp").

При этом три цветовых компонента размещаются один за другим в общем массиве **C** (рис. 5.4, *а)*. На рис. 5.4, *б* представлена графическая интерпретация массива **C** в виде изображения с градациями серого. Образ слева характеризует интенсивность красного в каждом пикселе изображения C.bmp, средний — интенсивность зеленого, а тот, который справа, — интенсивность синего.

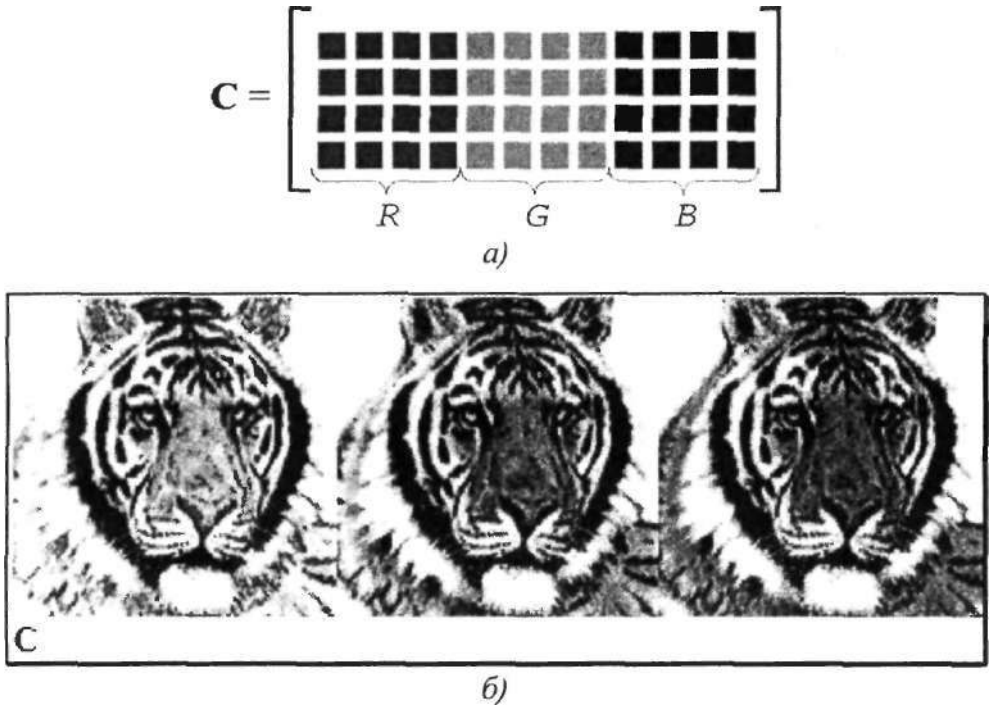



Рис. 5.4. Графическая интерпретация массива цветовых компонентов контейнера-оригинала

Для выделения цветовых составляющих можно использовать встроенные функции выделения соответствующих цветовых компонентов, каждая из которых возвращает массив, соответствующий определенному цветовому компоненту графического файла:

**R** := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

**Шаг 2**

В качестве сообщения, которое необходимо скрыть, используем, например, первые восемь абзацев из вступления данной книги. Текст сообщения сохраним в файле **M.txt** каталога, текущего для формируемого документа MathCAD, в следующем формате:

- тип файла — обычный текст (*.txt);
- кодирование — кириллица (Windows)[3].

[3] В принципе, существует возможность скрытия файлов любого формата. Единственное условие, которое при этом должно выполняться, — выбор файла-контейнера надлежащего объема (например, ориентировочное соотношение между объемами файла-контейнера и файла-сообщения для метода НЗБ — 8:1).

Импорт текстового сообщения можно выполнить с помощью функции **READBIN("имя_файла", "тип_формата_данных").** В этом случае данные представлены как 8-битное беззнаковое целое число (байт):

**M := READBIN**("M.txt", "byte").

Результат вычисления данного выражения — матрица-столбец (вектор), каждый элемент которой соответствует расширенному ASCII-коду соответствующего символа (буквы) импортируемого сообщения. В десятичном виде коды символов могут принимать значения от 0 до 255; в двоичном виде для этого достаточно использовать 8 битов на один символ — так называемое однобайтовое кодирование, на что указывает параметр byte как аргумент функции **READBIN** (см. приложения В и Е).

Фрагмент импортированного сообщения, а именно коды первых 16 символов (включая пробелы) в десятичном и двоичном видах, представлен на рис. 5.5[4].

Необходимо отметить, что по умолчанию нижняя граница индексации массивов равна 0. В примерах данной книги индексация начинается с 1 (если не будет оговорено другое), что, в частности, можно установить с помощью оператора **ORIGIN := 1** в начале документа или ввести 1 в поле **Array Origin** на вкладке **Built-in Variables** диалогового окна **Worksheet Options,** которое вызывается из меню **Tools** системы MathCAD.

Проверку импортирования файла сообщения (если в качестве сообщения используется обыкновенный текст) можно выполнить с помощью вызова функции **vec2str(M)**. Эта функция возвращает строку символов, соответствующих вектору M ASCII-кодов[5].

**Шаг 3**

Перед скрытием текстового файла **M.txt** в контейнере **C.bmp** его можно защитить криптографическим кодированием. Для наглядности и краткости изложения используем модифицированный код Виженера. Вообще, для устойчивости стегано-

[4] Для изменения формата отображенных данных в MathCAD необходимо выполнить следующее: выбрать операцию **Result** из меню **Format** (или дважды щелкнуть левой кнопкой мыши непосредственно на выведенных данных) для вызова диалогового окна **Result Format**; на вкладке **Display Options** в выпадающем меню **Radix** выбрать необходимую основу системы счисления: **decimal** (десятичная), **binary** (двоичная), **octal** (восьмеричная) или **hexadecimal** (шестнадцатеричная).

Возможна ситуация некорректного отображения кириллических символов в полученной строке. Эту проблему можно решить использованием кодовой страницы 1251 вместо 1250 и 1252, что можно задать заменой параметров 1250 и 1252 на параметр 1251 в реестре ОС Windows:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage;
STRING 1250 — значение c_1251.nls (если было c_1250.nls);
STRING 1252 — значение c_1251.nls (если было c_1252.nls).
```

За возможные проблемы, возникшие после редактирования реестра, авторы ответственности не несут. Для более безопасного редактирования реестра лучше воспользоваться специализированной программой, например, XP Tweaker RE. Кроме того, некорректность отображения кириллических символов в системе MathCAD никоим образом не влияет на результаты стеганографических преобразований, поскольку алгоритмы оперируют с двоичными отображениями ASCII-кодов, а извлеченное из контейнера сообщение можно без проблем прочитать в любом из распространенных текстовых редакторов.

сооощения к возможным искажениям рекомендуется применять код с исправлением ошибок.

M =

| | 1 |
|---|---|
| 1 | 200 |
| 2 | 237 |
| 3 | 244 |
| 4 | 238 |
| 5 | 240 |
| 6 | 236 |
| 7 | 224 |
| 8 | 246 |
| 9 | 232 |
| 10 | 255 |
| 11 | 32 |
| 12 | 255 |
| 13 | 226 |
| 14 | 235 |
| 15 | 255 |
| 16 | 229 |

M =

| | 1 |
|---|---|
| 1 | 11001000b |
| 2 | 11101101b |
| 3 | 11110100b |
| 4 | 11101110b |
| 5 | 11110000b |
| 6 | 11101100b |
| 7 | 11100000b |
| 8 | 11110110b |
| 9 | 11101000b |
| 10 | 11111111b |
| 11 | 100000b |
| 12 | 11111111b |
| 13 | 11100010b |
| 14 | 11101011b |
| 15 | 11111111b |
| 16 | 11100101b |

Рис. 5.5. Фрагмент сообщения, подлежащего скрытию

Алфавит источника сообщения зададим в виде ASCII-кодов: **i** := 1..256; $A_i$ := i–1 (для большей защищенности, элементы вектора **A** можно переставить по определенному закону, что соответствующим образом необходимо учитывать при расшифровывании).

Объем алфавита источника: **Na := rows(A)**, где **rows(A)** — функция, которая возвращает количество строк массива **A**. В данном случае: **Na** = 256 символов.

Из символов алфавита задаем секретный ключ, например: **K** := "@J|eKc-I98O". Количество символов в ключе (по соответствующей функции): **Nk := strlen(K)**, **Nk** = 11 символов.

Объем сообщения, которое подлежит кодированию: **Nm := rows(M)**, **Nm** = 5390 символов (включая скрытые служебные ASCII-символы переноса строки и возврата каретки — см. приложение Е).

Расширяем ключ на длину сообщения **(Nm)**, используя программный модуль (М.1).

$$
\begin{aligned}
K' := \;& K \leftarrow \mathbf{str2vec}(K) \\
& \textbf{for}\ \ i \in 1..Nm \\
& \quad r \leftarrow \mathbf{mod}(i, Nk) \\
& \quad K'_i \leftarrow K_r \quad \textbf{if}\ \ r > 0 \\
& \quad K'_i \leftarrow K_{Nk} \quad \textbf{if}\ \ r = 0 \\
& K'
\end{aligned}
\qquad (M.1)
$$

В данном модуле функция **str2vec(K)** преобразует строку символов **K** в вектор их ASCII-кодов. Программный оператор **for** организовывает цикл изменения **i** (переменная цикла) с заданным количеством повторов (в данном случае — от **1** до Nm). Функция **mod(i,Nk)** возвращает остаток от деления **i** на **Nk**.

Выполняем кодирование сообщения, используя модуль (М.2).

```
M_cod := | for j ∈ 1 .. Nm
         |   for i ∈ 1 .. Na
         |     m ← i  if  M_j = A_i
         |     n ← i  if  K'_j = A_i
         |   r ← mod(m + n , Na)
         |   M_cod_j ← A_r   if  r > 0
         |   M_cod_j ← A_Na  if  r = 0
         | M_cod
```

(М.2)

**Шаг 4**

Для того чтобы при распаковке контейнера из полученного множества символов можно было четко определить начало и конец именно скрытого сообщения, целесообразно ввести соответствующие секретные метки, которые ограничивали бы это полезное содержание.

Метки должны состоять из достаточного количества символов, чтобы не принимать за метки символы случайного образования. Кроме того, для уменьшения вероятности обнаружения меток при проведении стеганоанализа желательно, чтобы коды этих символов были достаточно разнесены на ASCII-оси (например, использовать наряду с латинскими символами символы кириллицы и служебных символов — так называемая *транслитерация;* использование псевдослучайных последовательностей кодов символов и т.п.). Пусть метки имеют следующий вид:

$$\mu_s := \text{"n0ч@m0к"}; \quad \mu_e := \text{"KiHeu,6"}.$$

Ограничивающие метки добавляем в текст закодированного сообщения, для чего используем функцию **stack(A,B,...)**, которая позволяет объединять записанные через запятую массивы. Объединение происходит путем "насаживания" матрицы А на матрицу **B**; полученной таким образом матрицы — на следующую матрицу (если такая присутствует) и т.д.

Понятно, что начальные матрицы должны иметь одинаковое количество столбцов, поэтому необходимо преобразовать метки из строк в векторы ASCII-кодов. Следовательно, **sMe := stack(str2vec($\mu_s$), M_cod, str2vec($\mu_e$))**.

Общее количество символов в скрытом сообщении: **rows(sMe) = 5404**. Количество НЗБ контейнера, которое для этого необходимо (8 бит/символ): **8·rows(sMe) = 43232** б и т а . О б щ е е количество НЗБ контейнера: **rows(C)·cols(C) = 3·128·128 = 49152 > 43232** бит. Таким образом, файл изображения имеет достаточный объем для того, чтобы скрыть сообщение.

**Шаг 5**

Для дальнейших вычислений потребуется преобразование десятичного числа (которым по умолчанию кодируется каждый символ) в формат двоичного. Также понадобится и обратное преобразование. Поскольку данные функции в MathCAD отсутствуют (существует, как это было показано выше, только возможность преобразования формата ответа, что для наших целей неприемлемо), предлагается использовать следующие модули, смысл которых совершенно очевиден.

Преобразование двоичного числа **x**, которое задано матрицей-столбцом (причем первый элемент матрицы — самый младший разряд числа), в десятичное выполняется с помощью модуля (М.3).

$$B2D(x) := \sum_{i=1}^{8} x_i \cdot 2^{i-1} \qquad (M.3)$$

Преобразование десятичного числа в двоичное реализуется модулем (М.4).

$$D2B(x) := \left| \begin{array}{l} \text{for } i \in 1..8 \\ \quad \left| \begin{array}{l} V_i \leftarrow \text{mod}(x, 2) \\ x \leftarrow \text{floor}\left(\frac{x}{2}\right) \end{array} \right. \\ V \end{array} \right. \qquad (M.4)$$

Функция **mod(x,2)** возвращает остаток от деления **x** на 2 (0, если **x** четное, и 1, если **x** нечетное). Функция **floor()** возвращает наибольшее целое число, которое меньше или равняется действительному значению аргумента.

**Шаг 6**

Для большего удобства и наглядности дальнейших действий, развернем матрицу **C** в вектор, временно изменив порядок цветовых матриц с **R-G-B** на **B-G-R**, что увеличит защищенность скрытой информации (на данном этапе можно использовать более надежные, но и более сложные, алгоритмы). В нашем случае применим модуль (М.5), в котором функция **augment(A,B,...)** объединяет матрицы **A, B, ...**, имеющие одинаковое количество строк (объединение выполняется столбец к столбцу, матрицы должны иметь одинаковое количество строк).

$$Cv := \left| \begin{array}{l} C' \leftarrow \text{augment}(B, G, R) \\ Cv \leftarrow C'^{(1)} \\ \text{for } i \in 2..\text{cols}(C') \\ \quad Cv \leftarrow \text{stack}\left(Cv, C'^{(i)}\right) \end{array} \right. \qquad (M.5)$$

Операция $C'^{(i)}$ позволяет выбирать **i**-й столбец из матрицы **C'**, каждый из которых впоследствии прибавляется к результирующему вектору **Cv**. Функция **cols(C')** возвращает общее количество столбцов массива **C'**.

**Шаг 7**

На основе вектора **Cv** формируем новый вектор, который уже будет содержать скрытое закодированное сообщение (модуль (М.6)).

Каждый символ закодированного сообщения (операция цикла **for** $\mu \in 1..$**rows(sMe)**) переводится в двоичный формат (переменная **b**), каждый из восьми разрядов которого записывается вместо НЗБ числа, соответствующего интенсивности того или иного цвета пикселя (последнее также предварительно переводится в двоичный формат (переменная **P**)).

После изменения модифицированное двоичное число **P** переводится в формат десятичного и записывается в соответствующую позицию вектора **Sv**

```
Sv := | for μ ∈ 1 .. rows(sMe)
      |    b ← D2B(sMe_μ)
      |    for i ∈ 1 .. 8
      |       P ← D2B[Cv_{i+8·(μ−1)}]
      |    * | P_1 ← b_i
      |       | Sv_{i+8·(μ−1)} ← B2D(P)
      | for j ∈ rows(Sv) − 1 .. rows(Cv)
      |       P ← D2B(Cv_j)
      |    * | P_1 ← round(rnd(1))
      |       | Sv_j ← B2D(P)
      | Sv
```

(M.6)

После обработки последнего символа сообщения **sMe** выполняется модификация элементов массива **Cv,** которые еще не претерпели изменений. Младшим битам каждого из таких элементов присваивается значение 0 или 1 (в данном случае — по равномерному закону распределения; функция **round()** возвращает округленное до ближайшего целого значение аргумента), хотя более правильно было бы провести исследование закона распределения значений уже модифицированных младших битов и соответствующим образом изменять те, которые остались (данная процедура является темой отдельного исследования и в рамках данной книги не рассматривается). Это делает невозможным со временем обнаружить факт модификации изображения.

В противном случае, проанализировав изображение, построенное из одних только НЗБ контейнера, нарушитель в большинстве случаев (если символов сообщения "не хватило" на весь контейнер) обнаружит границу ввода данных и при определенных усилиях сможет добыть скрытую информацию. Обычно, эту информацию еще необходимо расшифровать, но факт ее наличия уже будет раскрыт и вопрос защиты вернется к криптографической устойчивости используемого кодирования.

На рис. 5.6 в качестве примера представлены графические интерпретации массивов цветовых компонентов, воспроизведенные только на основе НЗБ (0/1) контейнера-оригинала *(а),* контейнера-результата без модификации (б) и с модификацией (в) избыточных битов. Как видим, при отсутствии "дописывания" при неполном заполнении контейнера четко прослеживается граница ввода сообщения (рис. 5.6, *б)* — следует напомнить, что подмассивы **R** и **B** поменялись местами, поэтому последней модифицировалась матрица **R,** которая, в конце концов, и оказалась заполненной не до конца. "Дописывание" равновероятными 0 и 1 несколько исправляет ситуацию, хотя при дополнительном стеганоанализе на закон распределения значений НЗБ несоответствие сразу будет обнаружено, что еще раз подтверждает необходимость предварительного анализа распределения значений уже модифицированных битов или же хотя бы продублировать часть сообщения для заполнения всего контейнера. Также очевидно отличие между рисунками *а* и *б.*

Таким образом, желательно, чтобы изображение, которое планируется использовать в качестве контейнера, было уникальным. Тут следует заметить, что все графические контейнеры условно разделяются на "чистые" и "зашумленные" [95].

У первых прослеживается связь между младшими и остальными семью битами цветовых компонентов, а также зависимость между самыми младшими битами.

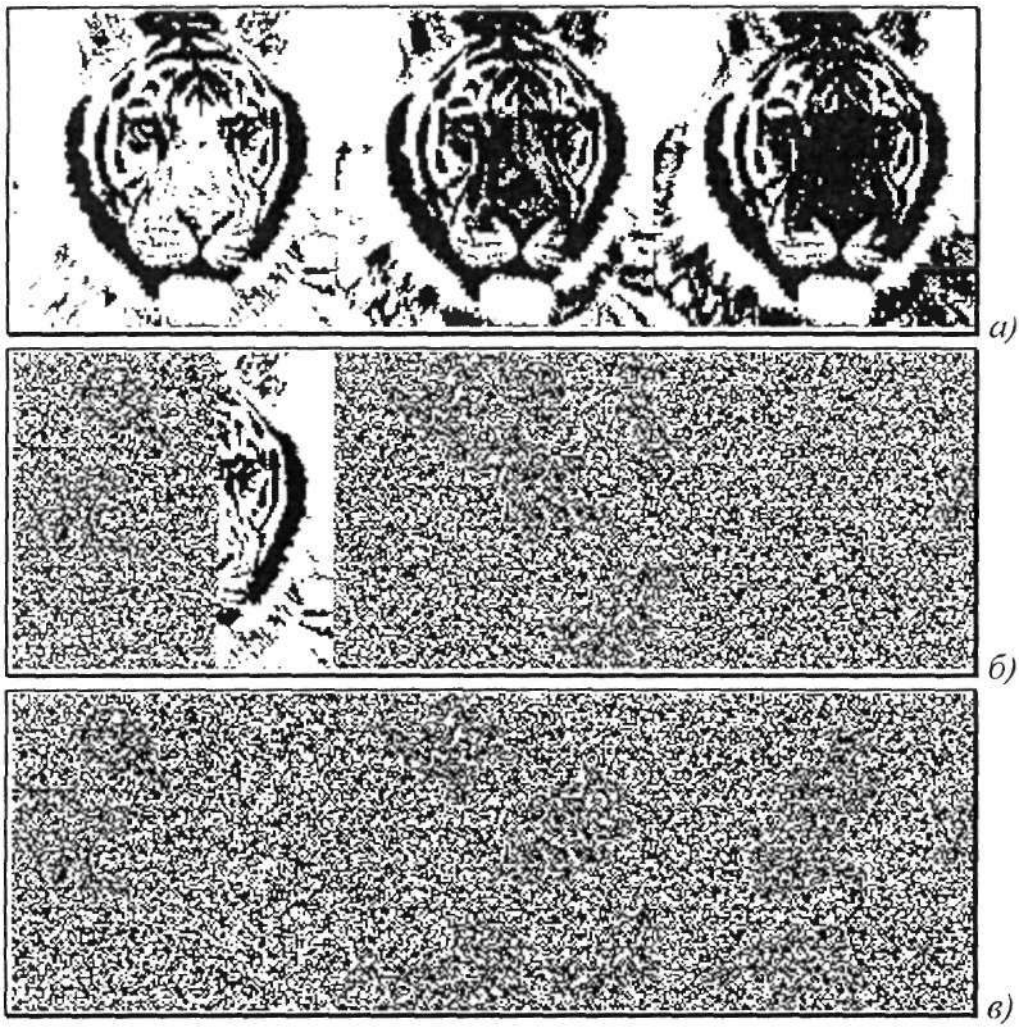



**Рис. 5.6. Массивы цветовых компонентов, воспроизведенных по НЗБ**

Встраивание сообщения в "чистое" изображение разрушает существующие зависимости, что, как было показано выше, легко обнаруживается. Если же изображение изначально зашумлено (сканированное изображение, цифровое фото и т.д.), то определить постороннее вложение становится на порядок сложнее, хотя и возможно при использовании теории вероятностей и математической статистики.

Также можно отметить, что для большей скрытности биты сообщения следует вносить не последовательно, а только в каждый второй или даже третий пиксель, или же подчинить внесение определенному, известному только авторизованным лицам, закону. Данные модификации легко осуществить путем внесения соответствующих незначительных изменений в модуль (М.6).

**Шаг 8**

Полученный с помощью модуля (М.6) вектор **Sv** сворачиваем в матрицу **S,** имеющую размерность первичной матрицы **C** (модули (М.7) и (М.8)).

$$S' := \text{for } i \in 1,2..\text{cols}(C) \quad S'^{(i)} \leftarrow \text{submatrix}[Sv,(i-1)\cdot\text{rows}(C)-1,i\cdot\text{rows}(C),1,1] \qquad \text{(М.7)}$$

Функция **submatrix(A, x, X, y, Y)** возвращает часть матрицы **A,** которая состоит из элементов, общих для строк от **x** до **X** и столбцов от **y** до **Y** включительно.

**Шаг 9**

Пользуясь этой же функцией, выделяем из массива **S'** цветовые матрицы и расставляем их на свои места (**R↔B**), получая контейнер-результат **S** (М.8).

$$\begin{aligned} \text{Вм} &:= \text{submatrix}\left(S', 1, \text{rows}(C), 1, \frac{\text{cols}(C)}{3}\right) \\ \text{Gм} &:= \text{submatrix}\left(S', 1, \text{rows}(C), \frac{\text{cols}(C)}{3} - 1, 2\cdot\frac{\text{cols}(C)}{3}\right) \\ \text{Rм} &:= \text{submatrix}\left(S', 1, \text{rows}(C), 2\cdot\frac{\text{cols}(C)}{3} - 1, \text{cols}(C)\right) \\ S &:= \text{augment}(\text{Rм}, \text{Gм}, \text{Вм}) \end{aligned} \qquad \text{(М.8)}$$

На рис. 5.7 показана графическая интерпретация массива **S** в виде изображения с градациями серого и воспроизведенное по цветовым составляющим изображение-контейнер со скрытым сообщением.

Сравнивая рис. 5.7 с рис. 5.4, можно сделать вывод об отсутствии заметных визуальных отклонений.

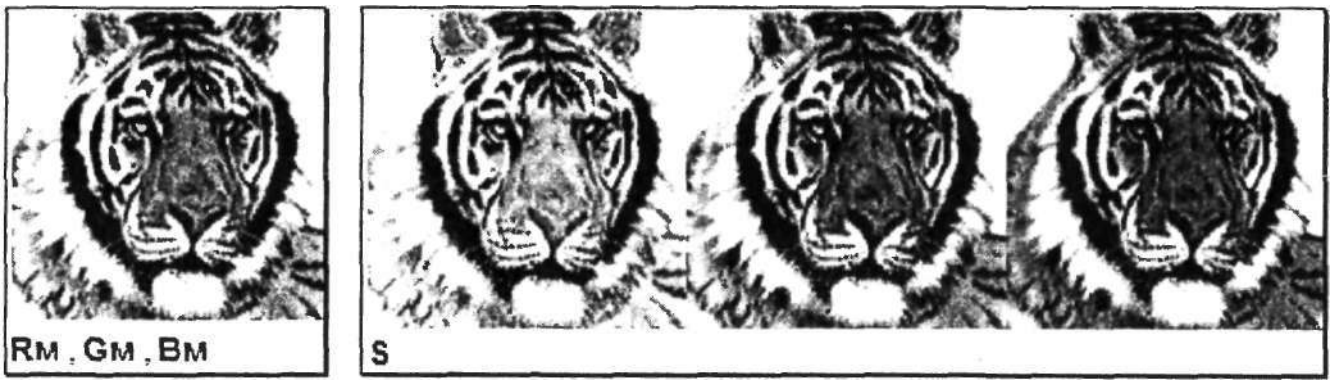



Рис. 5.7. Контейнер-результат и его интерпретация в виде массива цветовых компонентов

Остается только записать массив **S** в файл: **WRITERGB("S_LSB.bmp") := S**. Совершенно очевидно, что объем полученного файла будет соответствовать объему файла изображения-оригинала.

**Шаг 10**

Для исследования влияния на степень скрытости того, в какой из разрядов числа, характеризующего то или иное свойство пикселя (в нашем случае — интенсивность определенного цвета), будет заноситься секретная информация, в модуле (М.6) в отмеченных звездочкой строках следует вместо индекса " 1 " ввести индекс, соответствующий модифицируемому разряду. На рис. 5.8 изображен результат, полученный при внесении данных в 8-й (самый старший) бит числа, соответствующего интенсивности цветов.

Установлено, что визуально незаметно, если в качестве "носителей" использовать не только самый младший, но и следующий за ним бит каждого из указанных выше чисел (особенно если в изображении отсутствуют большие однотонные участки). Следовательно, в качестве одной из возможных степеней защиты возможно

использование изменяемой по определенному закону поочередной записи в эти *два* бита. Или же, жертвуя скрытностью, можно вдвое увеличить емкость контейнера.

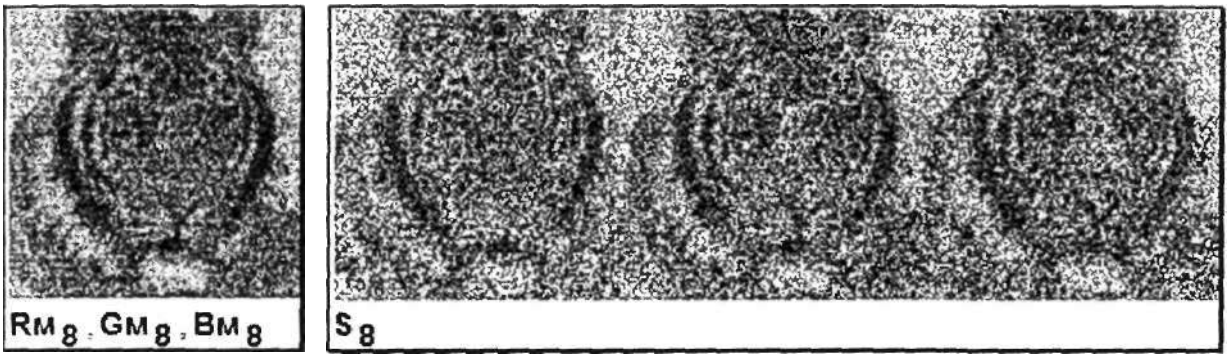



**Рис. 5.8. Цветное изображение и массив цветовых составляющих в случае внесения скрываемых данных в 8-й бит интенсивности цветов**

**Шаг 11**

Рассмотрим процесс распаковки скрытого сообщения. Изначально зная, что сообщение было помещено в массив цветовых компонентов, выделяем соответствующие каждому цвету подмассивы, переводя значения цветовых характеристик каждого пикселя изображения, содержащего в себе скрытое закодированное сообщение, в числовые матрицы[6]:

$$R^* := \text{READ\_RED}(\text{"S\_LSB.bmp"});\quad G^* := \text{READ\_GREEN}(\bullet);\quad B^* := \text{READ\_BLUE}(\bullet).$$

Из полученных матриц, соответствующим образом изменяя их порядок, аналогично (М.5), формируем вектор $Sv^*$ (модуль (М.9)).

$$\begin{aligned} Sv^* := \;& S' \leftarrow \text{augment}(B^*, G^*, R^*) \\ & Sv^* \leftarrow S'^{(1)} \\ & \text{for } i \in 2..\text{cols}(S') \\ & \quad Sv^* \leftarrow \text{stack}\left(Sv^*, S'^{(i)}\right) \end{aligned} \qquad \text{(М.9)}$$

Надлежащим образом обрабатывая каждые восемь элементов полученного вектора, распаковываем скрытое сообщение, используя модуль (М.10).

$$\begin{aligned} Mf^* := \;& \text{for } \mu \in 1..\frac{\text{rows}(Sv^*)}{8} \\ & \quad \text{for } i \in 1..8 \\ & \quad\quad P \leftarrow \text{D2B}\left[Sv^*_{i+8\cdot(\mu-1)}\right] \\ & \quad\quad b_i \leftarrow P_1 \\ & \quad Mf^*_\mu \leftarrow \text{B2D}(b) \\ & \quad Mf^*_\mu \leftarrow Mf^*_\mu - 32.5 \text{ if } Mf^*_\mu < 32 \\ & Mf^* \end{aligned} \qquad \text{(М.10)}$$

[6] Здесь и далее символом **"*"** будут помечаться параметры, которые относятся к принимающей стороне. Также по умолчанию будем считать, что принимающая сторона владеет алгоритмом встраивания.

Следует отметить, что, поскольку заранее неизвестно, какую часть вектора **Sv*** займет именно полезная информация, во внимание берутся все его элементы. Значения каждого элемента формируемого при этом вектора **Mf*** представляют собой коды символов "квазисообщения", которые вычисляются в обратном к (М.6) порядке: каждый младший разряд восьмерки преобразованных в двоичный формат элементов вектора **Sv*** формирует двоичное число кода символа, формат которого далее преобразуется в десятичный. Полученное число присваивается μ-му элементу вектора **Mf***.

**Шаг 12**

В связи с невозможностью обработки строковыми функциями 12-й версии MathCAD символов, ASCII-коды которых имеют значения от 0 до 31 включительно (за исключением служебных символов LF (код 10) и CR (код 13)), дополнительно вносится замена значений 0, 1, 2,..., 31 элементов вектора **Mf*** соответствующим прибавлением к каждому из них коэффициента 32,5 (коэффициент является дробью для того, чтобы в дальнейшем было возможно отличить "настоящие" значения элементов массива от тех, которые перед этим имели неформатные значения). Такую замену, конечно, необходимо соответствующим образом учитывать в дальнейшем — для этого запомним номера строк вектора **Mf***, элементы которых имеют дробные значения (модуль (М. 11)).

```
N := | for s ∈ 0, 1 .. 31
     |   i ← 1
     |   for μ ∈ 1 .. rows(Mf*)
     |     if Mf*_μ = s - 32.5
     |       N_{i, s+1} ← μ
     |       i ← i - 1
     | N
```
(М.11)

При этом, в первый столбец формируемого массива **N** заносятся номера элементов, значения которых равны 32,5 (бывшие нули), во второй столбец — номера элементов со значением 33,5 (бывшие единицы) и т.д.

**Шаг 13**

Зная, что текст полезной информации ограничен метками $\mu^*_s$ := "n0ч@m0к"; $\mu^*_e$ := "KiHeu,6", выделяем его из извлеченного квазисообщения, используя модуль (М.12).

Вектор ASCII-кодов **Mf***, предварительно преобразованный с помощью функции **vec2str()** в соответствующую ему строку символов, последовательно просматривается в поиске стартовой и конечной меток. Такая операция выполняется путем сравнения выделенной части строки данных с соответствующими метками, которые должны быть известны получателю.

Выделение выполняется с помощью функции **substr(Mf*, μ, β)**, которая возвращает подстроку длиной в β символов из строки **Mf***, начиная с символа μ (следует отметить, что в данном случае первый символ строки имеет номер, соответствующий значению встроенной переменной **ORIGIN**)[7]. В нашем случае, поскольку каждая из меток состоит из 7 символов, β = 7.

[7] В меню **Tools ► Worksheet Options,** на вкладке **Calculation** следует установить флажок **Use ORIGIN for string indexing.** Данная возможность отсутствует в предыдущих версиях MathCAD, в которых первый символ строки всегда имел индекс 0. Для корректной работы модуля (М.12) в этих версиях необходимо выполнить указанные изменения μ на μ – 1 и s на s – 1

```
M_cod* := | s ← 0
          | e ← 0
          | β_s ← strlen(μ*_s)
          | β_e ← strlen(μ*_e)
          | Mf* ← vec2str(Mf*)
          | for μ ∈ 1..strlen(Mf*)
          |     | s ← μ − β_s  if substr(Mf*, μ, β_s) = μ_s ∧ s = 0
          |     | e ← μ − 1    if substr(Mf*, μ, β_e) = μ_e ∧ e = 0
          |     | break if s ≠ 0 ∧ e ≠ 0
          | Mf* ← substr(Mf*, s, e − β_s)
          | M_cod* ← str2vec(Mf*)
          | for n ∈ 1..cols(N)
          |     for i ∈ 1..rows(N)
          |         | break if N_i,n = 0
          |         | M_cod*_(N_i,n − β_s) ← n − 1  if 0 < N_i,n ≤ rows(M_cod*) − β_s
          | M_cod*
```

(M.12)

Последовательным увеличением **μ**, происходит продвижение вдоль строки данных **Mf***. При выполнении указанных в модуле условий (оператор **if**), коэффициентам **s** и **e** присваиваются соответствующие значения номеров начальной и конечной позиции полезной информации в строке данных **Mf***. Такие дополнительные условия, как **s = 0**; **e = 0**; **s ≠ 0**; **e ≠ 0** введены для ускорения поиска.

Обратное преобразование строки символов **Mf*** в вектор их ASCII-кодов позволяет в этом же модуле непосредственно выделить скрытую информацию и восстановить элементы, значения которых были принудительно изменены на дробные.

**Шаг 14**

Распакованное сообщение требуется декодировать. Необходимые начальные условия состоят в том, что авторизованной стороне известны:

- алфавит источника с о о б (**i := 1..256; A*_i := i – 1**) м о м **Na* := rows(A*)** **Na* = 256** символов;
- секретный ключ **K* := "@J|eKc-I98O"** с **Nk* := strlen(K*)** **Nk* = 11** символов;
- объем сообщения, подлежащего декодированию: **Nm* := rows(M_cod*)**, **Nm* = 5390** символов.

С помощью модуля (М.13) секретный ключ **K*** расширяется на длину **Nm*** сообщения **M_cod*** (аналогично тому, как это было при кодировании сообщения — (М. 1)).

```
K** := | K* ← str2vec(K*)
       | for i ∈ 1..Nm*
       |     | r ← mod(i, Nk*)
       |     | K**_i ← K*_r   if r > 0
       |     | K**_i ← K*_Nk* if r = 0
       | K**
```

(M.13)

Декодирование секретного сообщения выполняется с помощью модуля (М.14).

$$
\begin{array}{l}
M^* := \left| \begin{array}{l}
\text{for } j \in 1 .. Nm^* \\
\quad \left| \begin{array}{l}
\text{for } i \in 1 .. Na^* \\
\quad \left| \begin{array}{l} m \leftarrow i \text{ if } M\_cod^*_j = A^*_i \\ n \leftarrow i \text{ if } K^*_j = A^*_i \end{array} \right. \\
r \leftarrow \mathrm{mod}(Na^* - m - n, Na^*) \\
M^*_j \leftarrow A^*_r \text{ if } r > 0 \\
M^*_j \leftarrow A^*_{Na^*} \text{ if } r = 0
\end{array} \right. \\
M^*
\end{array} \right.
\end{array}
\qquad (M.14)
$$

Декодированное сообщение записываем в файл:

**WRITEBIN**("M_dec.txt","byte",0) := **M***.

**Шаг 15**

Вычислим показатели визуального искажения, перечисленные в главе 3 (формулы (3.1)-(3.17)). Полученные результаты сведены в табл. 5.1 (стр. 125).

### 5.3.2.2. Метод псевдослучайного интервала

В рассмотренном выше простейшем случае выполняется замена НЗБ всех последовательно размещенных пикселей изображения. Другой подход — *метод случайного интервала* [78], заключается в случайном распределении битов секретного сообщения по контейнеру, в результате чего расстояние между двумя встроенными битами определяется псевдослучайно. Эта методика особенно эффективна в случае, когда битовая длина секретного сообщения существенно меньше количества пикселей изображения.

Рассмотрим простейший случай этого метода, когда интервал между двумя последовательными встраиваниями битов сообщения является функцией координат предыдущего модифицированного пикселя.

**Шаг 1**

Пусть сообщение, которое необходимо скрыть: **M** := "© Пузыренко А.Ю., 2005 г.". В качестве контейнера **C** используем подмассив **B** синего цветового компонента изображения рис. 5.3.

**Шаг 2**

Определим метки, которые будут устанавливать границы полезного сообщения в контейнере. В отличие от предыдущего метода, стартовая метка будет определять порядковый номер элемента контейнера, начиная с которого в последний будут заноситься данные. Пусть $\mu_s := 154$. Метка $\mu_e$ будет сигнализировать о завершении полезной части среди извлеченных символов: $\mu_e :=$ "KiHeu,6".

**Шаг 3**

Примем, что при внесении битов сообщения в контейнер со сменным шагом, величина последнего обусловлена количеством единиц в двоичном значении номера элемента контейнера, который модифицировался предварительно. Для подсчета ве-

личины шага (интервала) воспользуемся модулем (М.15), суммирующим количество элементов матрицы-столбца **x**, значение которых равно 1.

$$\mathbf{step}(x) := K \cdot \sum_{i=1}^{\mathbf{rows}(x)} x_i \qquad \text{(М.15)}$$

Коэффициент **K** в данном случае выступает в роли простейшего ключа, который может принимать любые целые значения (в том числе и отрицательные, но в этом случае стартовая метка должна иметь значение, близкое к наибольшему значению индекса элементов контейнера). Также при выборе **K** следует принимать во внимание общее количество бит, необходимых для скрытия сообщения, а также имеющееся количество элементов массива контейнера. Пусть **K** := 9.

Ограничивающую метку $\mu_e$ с помощью строчной функции **concat()**, объединяющей строки, выступающие в качестве ее аргументов, прибавим к тексту сообщения, которое подлежит скрытию. Результат объединения преобразуем в вектор ASCII-кодов: **Me** := **str2vec(concat(M**, $\mu_e$**))**. Общее количество символов в полученном сообщении: **rows(Me)** = 32 . Количество НЗБ контейнера, которое для этого необходимо (8 бит/символ): **8·rows(Me)** = 256 бит.

**Шаг 4**

Развернем массив **B** в вектор (М.16), на основе которого сформируем новый вектор, содержащий скрытое сообщение (М.17).

```
Cv := | Cv ← B^(1)
      | for i ∈ 2.. cols(B)
      |   Cv ← stack(Cv, B^(i))
```
(М.16)

```
Sv := | Sv ← Cv
      | z ← μ_s
      | for μ ∈ 1.. rows(Me)
      |   | b ← D2B(Me_μ)
      |   | for i ∈ 1.. 8
      |   |   | z ← z + step(D2B(z))
      |   |   | P ← D2B(Cv_z)
      |   |   | P_1 ← b_i
      |   |   | Sv_z ← B2D(P)
      | Sv
```
(М.17)

Каждый символ сообщения **Me** (операция цикла **for** $\mu \in$ **1..rows(Me)**) переводится в двоичный формат (переменная **b**), каждый разряд которого записывается вместо самого младшего бита числа **P**, соответствующего значению интенсивности синего цвета определенного пикселя. При этом элементы массива **Cv** перебираются не последовательно, а с переменным шагом, величина которого обусловлена функцией **step(•)**[8].

[8] При вычислении функции **D2B(z)** необходимо в (М.4) граничное значение переменной цикла **i** изменить с 8 на такое, которое позволит переводить в двоичный формат самый старший индекс элементов вектора **Cv**.

Стартовый элемент задается меткой $\mu_s$. После проведенного изменения, модифицированное двоичное число **P** переводится в формат десятичного и записывается в соответствующую позицию вектора **Sv**, который в начале модуля был принят равным вектору **Cv**.

**Шаг 5**

Обратное свертывание вектора **Sv** в массив, имеющий размерность контейнера, выполняется с использованием модуля (М.7) с тем лишь отличием, что аргументом функций размерности массива **(rows** и **cols)** является массив **B**. Чтобы оценить степень «рассеяния» бит скрытого сообщения по массиву контейнера, в качестве примера рассмотрим результат присвоения пикселю, в который планировалось ввести бит сообщения, нулевого значения интенсивности (черный цвет) при предварительном общем осветлении изображения (рис. 5.9).

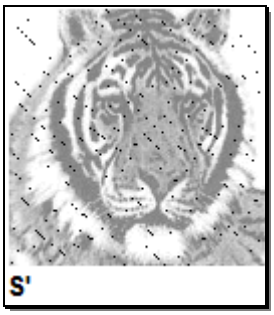



**Рис. 5.9. Рассеяние битов сообщения по массиву контейнера**

**Шаг 6**

Результирующее цветное изображение будет определяться массивом объединения цветовых массивов:

**S** := **augment**(**R**, **G**, **S'**).

**Шаг 7**

При извлечении скрытого сообщения должны быть известны параметры $\mu^*_s$, $\mu^*_e$, **K*** и, разумеется, массив **B***, который, как предполагается, содержит скрытые данные.

Развертывание массива **B*** в вектор **Sv*** выполняется с помощью аналогичного (М.16) модуля. Извлечение сообщения из вектора **Sv*** выполняется с помощью модуля (М.18) в обратном, по отношению к операции встраивания порядке.

```
Mf* := | z ← μ*_s
       | for μ ∈ 1 .. rows(Sv*)
       |     for i ∈ 1 .. 8
       |         z ← z + step(D2B(z))
       |         break if z > rows(Sv*)
       |         P ← D2B(Sv*_z)
       |         b_i ← P_1
       |     Mf*_μ ← B2D(b)
       |     μ ← μ + 1
       | Mf*
```

(М.18)

Из полученного вектора **Mf*** путем сравнения с меткой $\mu^*_e$ выделенного фрагмента извлекается полезное сообщение **M*** (М.19).

**M*** = "© Пузыренко А.Ю., 2005 г.".

Результаты вычисления визуального искажения объединены в табл. 5.1 (стр. 125).

```
M* := | μ*_e ← str2vec(μ*_e)
      | β_e ← rows(μ*_e)                                                          (M.19)
      | for μ ∈ 1 .. rows(Mf*) − β_e
      |   M* ← submatrix(Mf*, 1, μ − 1, 1, 1)  if  submatrix(Mf*, μ, μ + β_e − 1, 1, 1) = μ*_e
      | vec2str(M*)
```

#### 5.3.2.3. Метод псевдослучайной перестановки

Недостатком метода псевдослучайного интервала является то, что биты сообщения в контейнере размещены в той же последовательности, что и в самом сообщении, и только интервал между ними изменяется псевдослучайно. Поэтому для контейнеров фиксированного размера более целесообразным является использование метода ***псевдослучайной перестановки*** (***выбора***) [79], смысл которого заключается в том, что генератор ПСЧ образует последовательность индексов $j_1, j_2, ..., j_{l_M}$ и сохраняет $k$-й бит сообщения в пикселе с индексом $j_k$.

Пусть $N$ — общее количество бит (самых младших) в имеющемся контейнере; $P^N$ — перестановка чисел $\{1, 2, ..., N\}$. Тогда, если у нас имеется для скрытия конфиденциальное сообщение длинной $n$ бит, то эти биты можно просто встроить вместо бит контейнера $P^N(1), P^N(2), ..., P^N(n)$.

Функция перестановки должна быть псевдослучайной, иными словами, она должна обеспечивать выбор бит контейнера приблизительно случайным образом. Таким образом, секретные биты будут равномерно распределены по всему битовому пространству контейнера. Однако при этом индекс определенного бита контейнера может появиться в последовательности более одного раза и в этом случае может произойти «пересечение» — искажение уже встроенного бита. Если количество бит сообщения намного меньше количества младших бит изображения, то вероятность пересечения является незначительной, и искаженные биты в дальнейшем могут быть восстановлены с помощью корректирующих кодов.

Вероятность, по крайней мере, одного пересечения оценивается как

$$p \approx 1 - \exp\left[-\frac{l_M \cdot (l_M - 1)}{2 \cdot l_C}\right], \quad l_M << l_C . \tag{5.1}$$

При увеличении $l_M$ и $l_C = const$ данная вероятность стремится к единице.

Для предотвращения пересечений можно запоминать все индексы использованных элементов $j_i$ и перед модификацией нового пикселя выполнять предварительную его проверку на повторяемость. Также можно применять генераторы ПСЧ без повторяемости чисел. Последний случай рассмотрим более подробно.

Для наших целей функция перестановки также зависит от секретного ключа $K$. При этом генератор псевдослучайной перестановки $P^N$ — это функция, которая для каждого значения $K$ вырабатывает различные псевдослучайные перестановки чисел $\{1, 2, ..., N\}$.

Обозначим через $P_K^N$ генератор перестановок с соответствующим ключом $K$. Если перестановка $P_K^N$ защищена по вычислению (то есть, взлом алгоритма требует неоправданно больших затрат вычислительных ресурсов нарушителя), то возможность раскрытия содержания или предположение самого только вида перестановок без владения информацией о секретном ключе $K$ практически равняется нулю.

Секретный генератор псевдослучайной перестановки (ГПСП) может быть эффективно реализован на основе генератора псевдослучайной функции (ГПСФ) [98],

который, как и ГПСП, вырабатывает различные, не подлежащие прогнозированию функции при каждом отдельном значении ключа; однако множество значений функций не должно равняться области ее определения. ГПСФ легко реализуется из секретной хэш-функции $\boldsymbol{H}$ путем объединения аргумента $\boldsymbol{i}$ с секретным ключом $\boldsymbol{K}$ и взятия от результирующей битовой строки функции $\boldsymbol{H}$:

$$\boldsymbol{f_K(i) = H(K \circ i)}, \tag{5.2}$$

где $\boldsymbol{K \circ i}$ — объединение (конкатенация) битовых строк $\boldsymbol{K}$ и $\boldsymbol{i}$; $\boldsymbol{f_K(i)}$ — результирующая псевдослучайная функция от $\boldsymbol{i}$, которая зависит от параметра $\boldsymbol{K}$.

Генератор Луби (Luby) и Рекоффа (Reckoff) построен следующим образом. Запись вида $\boldsymbol{a \oplus b}$ подразумевает побитовое сложение по модулю 2 аргумента $\boldsymbol{a}$ с аргументом $\boldsymbol{b}$, причем результат сложения имеет ту же размерность, что и $\boldsymbol{a}$.

Пусть $\boldsymbol{i}$ — строка двоичных данных длиной $2 \cdot \boldsymbol{l}$. Разделим $\boldsymbol{i}$ на две части: $\boldsymbol{x}$ и $\boldsymbol{y}$ длиной $\boldsymbol{l}$ каждая, а ключ $\boldsymbol{K}$ — на четыре части: $\boldsymbol{K}_1$, $\boldsymbol{K}_2$, $\boldsymbol{K}_3$, $\boldsymbol{K}_4$. Тогда

$$\boldsymbol{y = y \oplus f_{K_1}(x) = y \oplus H(K_1 \circ x)};$$
$$\boldsymbol{x = x \oplus f_{K_2}(y) = x \oplus H(K_2 \circ y)};$$
$$\boldsymbol{y = y \oplus f_{K_3}(x) = y \oplus H(K_3 \circ x)};$$
$$\boldsymbol{x = x \oplus f_{K_4}(y) = x \oplus H(K_4 \circ y)};$$

возврат $\boldsymbol{y \circ x}$.

Для каждого значения ключа $\boldsymbol{K}$ алгоритм возвращает псевдослучайную перестановку из чисел $\{1, ..., 2^{2 \cdot l}\}$. Луби и Рекофф показали, что предлагаемая перестановка является настолько же секретной, насколько и генератор ПСФ. Они также предложили простой алгоритм перестановки из $\{1, ..., 2^{2 \cdot l+1}\}$. Если значение функции $\boldsymbol{f_K}$ представляет собой достаточно длинные битовые последовательности, тот же эффект можно получить, приняв, что $\boldsymbol{y}$ — первые $\boldsymbol{l}$ бит строки $\boldsymbol{i}$, а $\boldsymbol{x}$ — последние $(\boldsymbol{l}+1)$ бит.

Представленная выше конструкция позволяет получить перестановку $\boldsymbol{P_K^{2^k}}$ из $\{1, ..., 2^{\boldsymbol{k}}\}$ для произвольного $\boldsymbol{k}$. Однако в случае, когда количество бит контейнера составляет $\boldsymbol{N}$, возникает необходимость перестановки $\boldsymbol{P_K^N}$ с $\{1, ..., \boldsymbol{N}\}$.

Преимущество метода, предложенного в [79], заключается в том, что существует возможность ограничиться только имеющимися для $\boldsymbol{P_K^N}$ аргументами. Пусть $\boldsymbol{k} = [\mathbf{log}_2(\boldsymbol{N})]$ (квадратные скобки означают округление к наименьшему целому, которое больше или равно аргументу). Тогда $2^{k-1} < \boldsymbol{N} \le 2^{\boldsymbol{k}}$. При этом подсчитываются значения $\boldsymbol{P_K^{2^k}}(1)$, $\boldsymbol{P_K^{2^k}}(2)$,… и из последовательности удаляются любые числа, превышающие $\boldsymbol{N}$. Таким образом, получают значения $\boldsymbol{P_K^N}(1)$, $\boldsymbol{P_K^N}(2)$, … . Заметим, что это становится возможным, когда функция перестановки вычислена для возрастающих значений аргументов, начиная с единицы. Следовательно, генератор ПСП $\boldsymbol{P^N}$ для произвольного $\boldsymbol{N}$ может быть построен на основе алгоритма Луби и Рекоффа.

Если же $\boldsymbol{N}$ является составным (как в случае изображения), существует более удобный способ построения ГПСП. Представленный ниже алгоритм основан на блочном кодировании с произвольным размером блока [79, 99]. Количество бит контейнера должно представлять собой составное число из двух сомножителей приблизительно одинакового порядка, то есть $\boldsymbol{N = X \cdot Y}$ для некоторых $\boldsymbol{X}$ и $\boldsymbol{Y}$.

В случае, когда данные скрываются в НЗБ пикселей цифрового изображения, параметры $X$ и $Y$ являются размерами данного изображения. Для получения координат $i$-го пикселя изображения при скрытии бита сообщения ($i \in \{1, ..., N\}$) необходимо выполнить следующие вычисления:

$$x = \mathbf{div}(i, Y) + 1; \quad (5.3\ a)$$

$$y = \mathbf{mod}(i, Y) + 1; \quad (5.3\ б)$$

$$x = \mathbf{mod}\left(x + f_{K_1}(y), X\right) + 1; \quad (5.3\ в)$$

$$y = \mathbf{mod}\left(y + f_{K_2}(x), Y\right) + 1; \quad (5.3\ г)$$

$$x = \mathbf{mod}\left(x + f_{K_3}(y), X\right) + 1; \quad (5.3\ д)$$

$$i = (x, y) \text{ или } i = (x - 1) \cdot Y + y, \quad (5.3\ е)$$

где $\mathbf{div}(i, X)$ и $\mathbf{mod}(i, X)$ — функции, которые возвращают, соответственно, целое и остаток от деления $i$ на $X$. Другой вариант формулы (5.3 *е*) применим в случае, если массив изображения предварительно был развернут в вектор (по строкам). Прибавление единицы необходимо при индексации элементов массива изображения, начиная с единицы.

Первые два раунда алгоритма ((5.3*а*) – (5.3*г*)) необходимы для того, чтобы «рассеять» биты скрываемого сообщения среди наименее значимых бит контейнера. При этом первый раунд придает случайный характер $x$-координатам пикселя-контейнера, а второй — $y$-координатам. Третий раунд необходим для противодействия атаке на открытый (незашифрованный) текст.

В случае использования только двух раундов, пусть $i = (b - 1) \cdot Y + a$, а $P_K^N(i)$ — значение перестановки. Если криптоаналитик способен предположить значение $a$ и может получить пару «открытый текст — кодированный текст» ($i' = (z - 1) \cdot Y + a$, $P_K^N(i')$) для некоторого $z$, то он способен установить $b$. Несмотря на то, что авторы [79] считают, что предложенный ими алгоритм будет достаточно устойчивым и с тремя раундами, они признают, что в некоторых случаях может понадобиться четыре или более раундов, что должно повысить устойчивость алгоритма к взлому.

Промоделируем данный метод в программе MathCAD.

**Шаг 1**

Сообщение, которое необходимо скрыть: **M** := "© Пузыренко А.Ю., 2005 г.".

Контейнер **C** — подмассив **B** синей цветовой составляющей изображения рис. 5.3. При этом количество бит в сообщении: $L_M$ := **strlen**(**M**), $L_M$ = 200 бит; геометрические размеры контейнера: **X** := **rows**(**C**), **X** = 128 пикселей; **Y** := **cols**(**C**), **Y** = 128 пикселей; **N** := **X**·**Y**, **N** = 16384.

**Шаг 2**

Для формирования ключа используем модуль (М.20), который позволяет на основе первичного ключа $K_O \geq 2$ сформировать вектор, который будет содержать $\Re$ пар ключей (каждая пара ключей будет использоваться в соответствующем раунде вычисления координат **x** и **y**).

В данном модуле поочередно используются три функции:

- **num2str**(**d**) — преобразование аргумента-числа в соответствующую строку **A**;
- **substr**(**A**, 0, 3) — выделение фрагмента строки **A**, который содержит три первых символа строки;

- **str2num**(**a**) — обратное преобразование **a**-фрагмента в число, которое и присваивается **s**-му элементу массива **K**.

```
K := | for s ∈ 1 .. 2·ℜ
     |   K_s ← K_0                                              if s = 1
     |   K_s ← str2num[substr[num2str[(K_{s-1})^2], 0, 3]]      if s > 1
     |   K_s ← str2num(substr(num2str(K_s), 0, 2))              if K_s > 255
     | K
```
(M.20)

В случае $K_s > 255$ с помощью аналогичной комбинации функций число сокращается на один символ.

Рассмотрим пример вычисления пар ключей при $K_o = 125$ и $\Re = 6$ (рис. 5.10).

$K^T =$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 125 | 156 | 243 | 59 | 34 | 115 | 132 | 174 | 30 | 90 | 81 | 65 |

**Рис. 5.10. Пример вычисления ключей K**

**Шаг 3**

Встраивание бит сообщения в псевдопиксели контейнера выполним с помощью модуля (М.21), который реализует алгоритм (5.3). В начале модуля массиву **S** присваиваются значения исходного массива **C**. Также выполняется конвертирование сообщения из строкового формата в вектор двоичных данных $M_{vec\_bin}$. При вычислении координат **x** и **y** используется операция векторизации, которая позволяет поэлементно складывать по модулю 2 двоичные векторы **K** и **y** (или **x**). При этом размерность указанных векторов должна быть одинаковой, для чего используется функция **submatrix**(•).

```
S := | S ← C
     | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. rows(M_vec)                                          } (I)
     |   M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | for i ∈ 1 .. L_M
     |   x ← floor(i / Y) + 1
     |   y ← mod(i, Y) + 1
     |   for s ∈ 1 .. ℜ
     |     x ← mod(x + B2D((D2B(K_{2·s-1}) ⊕ submatrix(D2B(y), 1, 8, 1, 1))→), X) + 1
     |     y ← mod(y + B2D((D2B(K_{2·s}) ⊕ submatrix(D2B(x), 1, 8, 1, 1))→), Y) + 1
     |   P ← D2B(C_{x,y})
     |   P_1 ← M_vec_bin_i
     |   S_{x,y} ← B2D(P)
     | S
```
(M.21)

Оценим степень «рассеяния» бит сообщения по массиву контейнера при разном количестве раундов $\Re$ вычисления координат **x** и **y** (рис. 5.11, формирование которого проводится аналогично формированию рис. 5.9).

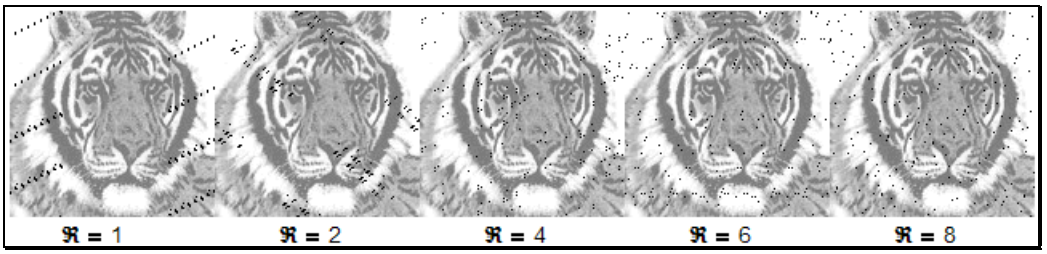



**Рис. 5.11. Рассеяние бит сообщения по массиву контейнера при изменении параметра $\Re$**

Как видим, приемлемый уровень рассеяния достигается уже при $\Re = 4$.

**Шаг 4**

На принимающей стороне должны быть известны первичный ключ $K_o^*$ и массив цветности, в который выполнялось встраивание (**S***). Из последнего получаются значения **X***, **Y***, **N***. Модуль, предназначенный для извлечения скрытого сообщения, представлен ниже (М.22).

```
M* := for i ∈ 1 .. N*/8
        x* ← floor(i/Y*) + 1
        y* ← mod(i, Y*) + 1
        for s ∈ 1 .. ℜ
          x* ← mod(x* + B2D((D2B(K*_(2·s−1)) ⊕ submatrix(D2B(y*), 1, 8, 1, 1))→, X*) + 1
          y* ← mod(y* + B2D((D2B(K*_(2·s)) ⊕ submatrix(D2B(x*), 1, 8, 1, 1))→, Y*) + 1
        P* ← D2B(S*_(x*, y*))
        M* vec_bin_i ← P*_1
      for j ∈ 1 .. rows(M* vec_bin)/8
        M* vec_j ← B2D(submatrix(M* vec_bin, 8·j − 7, 8·j, 1, 1))
      vec2str(M* vec)
```
(М.22)

Отметим, что в данном и последующих методах предварительное добавление стартовой и конечной меток к встраиваемому сообщению не выполняется. Соответственно, при извлечении строка символов, кроме полезного содержания, будет содержать еще и произвольный набор символов, входящих в кодировку ASCII.

Отсутствие в последующих модулях этапов выделения полезного сообщения из всего множества символов ни коим образом не говорит о невозможности осуществления данной операции. При желании, представленные модули можно легко адаптировать для осуществления поиска предварительно встроенных меток (по аналогии с рассмотренными выше методами).

Результаты вычисления визуального искажения сведены в табл.5.1 (стр.125).

#### 5.3.2.4. Метод блочного скрытия

***Метод блочного скрытия*** — это еще один подход к реализации метода замены и заключается в следующем [3]. Изображение-оригинал разбивается на $l_M$ непересекающихся блоков $\Delta_i$ ($1 \le i \le l_M$) произвольной конфигурации, для каждого из которых вычисляется бит четности $b(\Delta_i)$:

$$b(\Delta_i) = \sum_{j \in \Delta_i}^{\mathrm{mod}\,2} LSB(C_j). \qquad (5.4)$$

В каждом блоке выполняется скрытие одного секретного бита $M_i$. Если бит четности $b(\Delta_i) \neq M_i$, то происходит инвертирование одного из НЗБ блока $\Delta_i$, в результате чего $b(\Delta_i) = M_i$. Выбор блока может происходить псевдослучайно с использованием стеганоключа.

Хотя этот метод имеет такую же устойчивость к искажениям, как и все предыдущие, у него есть ряд преимуществ. Во-первых, существует возможность модифицировать значение такого пикселя в блоке, изменение которого приведет к минимальному изменению статистики контейнера. Во-вторых, влияние последствий встраивания секретных данных в контейнер можно уменьшить за счет увеличения размера блока.

Рассмотрим пример программы в MathCAD, которая позволяет выполнить стеганографическую защиту текстового сообщения методом блочного скрытия.

**Шаг 1**

Исходные данные соответствуют принятым при моделировании предыдущего метода.

**Шаг 2**

Разбиение массива контейнера на блоки выполним следующим образом: если количество бит в сообщении ($\mathbf{L_M}$) не превышает количества столбцов **Y** массива **C**, то один блок соответствует отдельному столбцу массива **C**. Если же $\mathbf{L_M} > \mathbf{Y}$, то один блок равен $1/\chi$ от отдельного столбца массива, где $\chi = \mathbf{ceil}(\mathbf{L_M} \div \mathbf{Y})$. Значение $\chi$ должно быть известно получателю.

Данный алгоритм встраивания реализован в модуле (М.23). Начало модуля аналогично модулю (М.21). Счетчик $\sigma$ позволяет выделять соответствующую соотношению $\chi$ часть от общей размерности столбца массива. При этом определяются индексы строк, начиная с которой (**r1**) и по которую (**r2**) выделяется фрагмент $\Delta$ **y**-го столбца.

Для каждого блока $\Delta$ выполняется вычисление бита четности **b**. Если **b** не равен текущему значению бита сообщения, то из блока $\Delta$ случайным образом выбирается индекс **n** пикселя, интенсивность цвета которого увеличивается или уменьшается на единицу, в зависимости от того, четным или нечетным является его первичное значение. С помощью функции **putregion**(**S**, $\Delta$, **r1**, **y**) выполняется встраивание модифицированного массива $\Delta$ в общий массив **S**, начиная со строки **r1** и столбца **y** в сторону самых старших индексов строк и столбцов соответственно.

На рис. 5.12 изображены пиксели контейнера, интенсивность цвета которых претерпела изменения (для наглядности, цвет данных пикселей установлен черным).

```
S := | S ← C
     | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. rows(M_vec)
     |   M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | for σ ∈ 1 .. χ
     |   | r1 ← (σ − 1)·floor(X ÷ χ) + 1
     |   | r2 ← σ·floor(X ÷ χ)
     |   | for y ∈ 1 .. Y
     |   |   | break if y + (σ − 1)·Y > L_M
     |   |   | Δ ← submatrix(S, r1, r2, y, y)
     |   |   | b ← 0
     |   |   | for x ∈ 1 .. rows(Δ)
     |   |   |   | P ← D2B(Δ_x)
     |   |   |   | LSB ← P_1
     |   |   |   | b ← b ⊕ LSB
     |   |   | if b ≠ M_vec_bin_{y+(σ−1)·Y}
     |   |   |   | n ← ceil(rnd(rows(Δ)))
     |   |   |   | Δ_n ← Δ_n + 1 if mod(Δ_n, 2) = 0
     |   |   |   | Δ_n ← Δ_n − 1 otherwise
     |   |   |   | S ← putregion(S, Δ, r1, y)
     | S
```

(M.23)

Очевидным является факт снижения количества модифицированных пикселей по сравнению с результатом применения двух предыдущих методов (см. рис. 5.9 и 5.11).

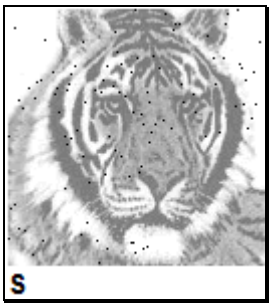



**Рис. 5.12. Рассеяние бит сообщения по массиву контейнера**

**Шаг 3**

Извлечение секретной информации осуществляется с помощью модуля (М.24).

#### 5.3.2.5. Методы замены палитры

Для скрытия данных можно также воспользоваться палитрой цветов, присутствующих в формате изображения [80]. Палитра из $N$ цветов определяется как список пар индексов ($i$, $\Lambda_i$), который определяет соответствие между индексом $i$ и его вектором цветности $\Lambda_i$ (так называемая таблица цветов). Каждому пикселю изображения ставится в соответствие определенный индекс в таблице. Поскольку порядок цветов в палитре не важен для восстановления общего изображения, конфиденциальная информация может быть скрыта путем перестановки цветов в палитре.

```
M* := | for σ ∈ 1 .. χ*
      |   r1 ← (σ − 1)·floor(X* ÷ χ*) + 1
      |   r2 ← σ·floor(X* ÷ χ*)
      |   for y ∈ 1 .. Y*
      |     Δ ← submatrix(S* , r1 , r2 , y , y)
      |     b ← 0
      |     for x ∈ 1 .. rows(Δ)
      |       P ← D2B(Δ_x)
      |       LSB ← P_1
      |       b ← b ⊕ LSB
      |     M* vec_bin_(y+(σ−1)·Y) ← b
      | for j ∈ 1 .. rows(M* vec_bin) ÷ 8
      |   M* vec_j ← B2D(submatrix(M* vec_bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M* vec)
```

(M.24)

Результаты вычисления визуального искажения сведены в табл. 5.1.

Существует $N!$ различных способов перестановки $N$-цветной палитры, чего вполне достаточно для скрытия небольшого сообщения. Однако методы скрытия, в основе которых лежит порядок формирования палитры, также являются неустойчивыми: любая атака, связанная со сменой палитры, уничтожает встроенное сообщение.

Чаще всего соседние цвета в палитре не обязательно похожи, поэтому некоторые стеганометоды перед скрытием данных упорядочивают палитру таким образом, что смежные цвета становятся подобными. Например, значение цвета может быть упорядочено по расстоянию $d$ в RGB-пространстве, где $d = \sqrt{R^2 + G^2 + B^2}$ [3].

Поскольку ЗСЧ более чувствительна к изменениям яркости цвета, то целесообразно сортировать содержание палитры именно по значениями яркости сигнала. После сортировки палитры можно изменять НЗБ индексов цвета без чрезмерного искажения изображения.

Некоторые стеганометоды [81] предусматривают уменьшение общего количества значений цветов (до $N/2$) путем «размывания» изображения. При этом элементы палитры дублируются таким образом, чтобы значение цветов для них различалось несущественно. В итоге каждое значение цвета размытого изображения соответствует двум элементам палитры, которые выбираются в соответствии с битом скрываемого сообщения.

Можно предложить следующий вариант метода замены палитры.

**Шаг 1**

Исходные данные — стандартные.

**Шаг 2**

Таблицу цветов получаем, используя подмассив интенсивности красного **R** (М.25). Секретность таблицы будет определять алгоритм ее формирования на основе массива **R**.

```
T := | i ← 1
     | for x ∈ 1 .. X
     |   for y ∈ 1 .. Y
     |     | break if i = 257
     |     | Λ ← R_{x,y}
     |     | if i = 1
     |     |   | T_{i,1} ← i
     |     |   | T_{i,2} ← Λ
     |     |   | i ← i + 1
     |     | if i > 1
     |     |   | δ ← 0
     |     |   | for j ∈ 1 .. i − 1
     |     |   |   δ ← 1 if T_{j,2} = Λ
     |     |   | if δ = 0
     |     |   |   | T_{i,1} ← i
     |     |   |   | T_{i,2} ← Λ
     |     |   |   | i ← i + 1
     | T
```

(M.25)

Полученную таблицу **T** упорядочим по интенсивности цветов с помощью функции **csort**(**T**, **c**), которая позволяет переставить строки массива **T** таким образом, чтобы отсортированным оказался столбец **c**:

$$\mathbf{T}_{\mathbf{sort}} := \mathbf{csort}(\mathbf{T}, 2).$$

На рис. 5.13 представлены фрагменты оригинальной и отсортированной цветовых таблиц.

**T** =

| | 1 | 2 |
|---|---|---|
| 1 | 1 | 255 |
| 2 | 2 | 251 |
| 3 | 3 | 250 |
| 4 | 4 | 249 |
| 5 | 5 | 247 |
| 6 | 6 | 244 |
| ⋮ | ⋮ | ⋮ |
| 251 | 251 | 230 |
| 252 | 252 | 75 |
| 253 | 253 | 196 |
| 254 | 254 | 193 |
| 255 | 255 | 55 |
| 256 | 256 | 2 |

$\mathbf{T}_{\mathbf{sort}}$ =

| | 1 | 2 |
|---|---|---|
| 1 | 99 | 0 |
| 2 | 49 | 1 |
| 3 | 256 | 2 |
| 4 | 98 | 3 |
| 5 | 139 | 4 |
| 6 | 174 | 5 |
| ⋮ | ⋮ | ⋮ |
| 251 | 3 | 250 |
| 252 | 2 | 251 |
| 253 | 74 | 252 |
| 254 | 108 | 253 |
| 255 | 109 | 254 |
| 256 | 1 | 255 |

**Рис. 5.13. Оригинальная (T) и отсортированная ($T_{sort}$) цветовые таблицы**

На рис. 5.14. представлена графическая интерпретация цветовых таблиц.


**Рис. 5.14. Графическое отображение цветовых таблиц T и $T_{sort}$**

**Шаг 3**

Модуль встраивания сообщения в контейнер (М.26) реализует следующий алгоритм.

```
S := | Ⓘ -- см. (М.21)
     | μ ← 1
     | for x ∈ 1..X
     |   for y ∈ 1..Y
     |     | break if μ > L_M
     |     | pix ← C_{x,y}
     |     | for i ∈ 1..256
     |     |   if T_sort_{i,2} = pix
     |     |     | n ← T_sort_{i,1}
     |     |     | ℑ ← i
     |     |     | break
     |     | if mod(n,2) ≠ M_vec_bin_μ
     |     |   | for σ ∈ 1..255
     |     |   |   | λ_L ← −1000
     |     |   |   | if mod(T_sort_{ℑ−σ,1},2) = M_vec_bin_μ if ℑ − σ ≥ 1
     |     |   |   |   | λ_L ← T_sort_{ℑ−σ,2}
     |     |   |   |   | break
     |     |   | for σ ∈ 1..255
     |     |   |   | λ_H ← 1000
     |     |   |   | if mod(T_sort_{ℑ+σ,1},2) = M_vec_bin_μ if ℑ + σ ≤ 256
     |     |   |   |   | λ_H ← T_sort_{ℑ+σ,2}
     |     |   |   |   | break
     |     |   | S_{x,y} ← λ_L if pix − λ_L ≤ λ_H − pix
     |     |   | S_{x,y} ← λ_H if pix − λ_L > λ_H − pix
     |     | μ ← μ + 1
     | S
```

(М.26)

Формирование битового вектора из символьной строки аналогично представленному в (М.21). Из массива контейнера **C**, путем перебора индексов строк (**x**) и столбцов (**y**), переменной **pix** присваивается значение интенсивностей цвета соответствующих пикселей контейнера. Внутренним циклом **i** ∈ 1..256 выполняется по-

иск соответствующего значения интенсивности в отсортированной цветовой таблице $\mathbf{T_{sort}}$.

В случае нахождения, переменной **n** присваивается значение индекса, соответствующего данной интенсивности в таблице **T** (первый столбец $\mathbf{T_{sort}}$), а переменной $\mathfrak{I}$ — значение индекса, соответствующего данной интенсивности в таблице $\mathbf{T_{sort}}$. Если НЗБ индекса **n** не равно текущему биту скрываемого сообщения, то происходит поиск ближайшего индекса, НЗБ которого равно биту сообщения. Поиск выполняется вниз (**L**) и вверх (**H**) от индекса $\mathfrak{I}$.

Предварительное присвоение переменным $\lambda_L$ и $\lambda_H$ значения $-/+1000$ гарантирует невозможность дублирования предыдущих значений $\lambda$, если продвижение вниз или вверх от индекса $\mathfrak{I}$ не привело к выполнению поставленного условия (последнее возможно при обнаружении индекса $\mathfrak{I}$ слишком близко к нижней или верхней границе отсортированной цветовой таблицы).

После того как значения $\lambda_L$ и $\lambda_H$ найдены, пикселю контейнера **S** присваивается то из них, которое на цветовой оси находится ближе к интенсивности **pix** соответствующего пикселя контейнера **C**. После встраивания последнего бита сообщения, внешний цикл прерывается — контейнер заполнен.

#### Шаг 4

При извлечении сообщения, на основе массива **R*** необходимо сформировать таблицы цветов **T** и $\mathbf{T_{sort}}$. Модуль, реализующий данную операцию, идентичен (М.25).

#### Шаг 5

Модуль извлечения (М.27) для интенсивности каждого пикселя массива **S*** выполняет поиск соответствующей интенсивности в цветовой таблице. При нахождении, $\mu$-му элементу битового сообщения **M*** присваивается значение НЗБ индекса, соответствующее данной интенсивности в неотсортированной таблице. Полученный битовый вектор в конце модуля преобразуется в строку символов.

```
M* := | μ ← 1
      | for x ∈ 1 .. X*
      |   for y ∈ 1 .. Y*
      |     | pix ← S*x,y
      |     | for i ∈ 1 .. 256
      |     |   if T* sorti,2 = pix
      |     |     | M*μ ← mod(T* sorti,1, 2)
      |     |     | μ ← μ + 1
      | for j ∈ 1 .. rows(M*) ÷ 8
      |   M* vecj ← B2D(submatrix(M*, 8·j − 7, 8·j, 1, 1))
      | vec2str(M* vec)
```

(М.27)

Результаты вычисления визуального искажения сведены в табл. 5.1.

#### 5.3.2.6. Метод квантования изображения

К методам скрытия в пространственной области можно также отнести метод ***квантования изображения*** [3, 82], основанный на межпиксельной зависимости, которую можно описать некоторой функцией $\Theta$. В простейшем случае можно вычислить разницу $\varepsilon_i$ между смежными пикселями $\mathbf{c}_i$ и $\mathbf{c}_{i+1}$ (или $\mathbf{c}_{i-1}$ и $\mathbf{c}_i$) и задать ее как параметр функции $\Theta$: $\Delta_i = \Theta(\mathbf{c}_i - \mathbf{c}_{i+1})$, где $\Delta_i$ — дискретная аппроксимация разницы сигналов $\mathbf{c}_i - \mathbf{c}_{i+1}$.

Поскольку $\Delta_i$ — целое число, а реальная разница $\mathbf{c}_i - \mathbf{c}_{i+1}$ — действительное число, то возникают ошибки квантования $\delta_i = \Delta_i - \varepsilon_i$. Для сильно коррелированных сигналов эта ошибка близка к нулю: $\delta_i \approx 0$.

При данном методе скрытие информации производится путем корректировки разностного сигнала $\Delta_i$. Стеганоключ представляет собой таблицу, которая каждому возможному значению $\Delta_i$ ставит в соответствие определенный бит, например:

| $\Delta_i$ | –4 | –3 | –2 | –1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| $b_i$ | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Для скрытия $i$-го бита сообщения вычисляется разница $\Delta_i$. Если при этом $b_i$ не соответствует секретному биту, который необходимо скрыть, то значение $\Delta_i$ заменяется ближайшим $\Delta_j$, для которого такое условие выполняется. При этом соответствующим образом корректируются значения интенсивностей пикселей, между которыми вычислялась разница $\Delta_i$. Извлечение секретного сообщения осуществляется согласно значению $b^*_i$, соответствующему разнице $\Delta^*_i$.

Рассмотрим пример программы, реализующей метод квантования изображения.

**Шаг 1**

Исходные данные — стандартные.

**Шаг 2**

Стеганоключ вычисляем по модулям (М.28) и (М.29). При этом модуль (М.28) возвращает все возможные разницы сигналов (от –255 до +255), а модуль (М.29) — значения бит, соответствующие этим разницам.

```
Δ := | for i ∈ 1 .. 511
     |   Δ_i ← i − 256
     | Δ
```
(М.28)

```
b := | i ← 1
     | for y ∈ 1 .. Y
     |   | c ← R^⟨y⟩
     |   | b_i ← 0
     |   | for x ∈ 1 .. X
     |   |   | b_i ← b_i ⊕ mod(c_x , 2)
     |   |   | b_i ← b_i ∨ 1  if mod(x , 3) = 0
     |   | i ← i + 1
     | b ← stack(b , b , b , b)
```
(М.29)

Значения $b_i$ в данном случае рассчитываются на основании массива красной цветовой составляющей. При этом для каждой колонки массива **R** рассчитывается сумма по модулю 2 составляющих ее элементов с булевым прибавлением к результату суммирования единицы при каждом третьем элементе. В конце модуля полученный вектор **b** расширяется на длину вектора Δ. Таким образом, элементы массива **b** носят псевдослучайный характер. Фрагменты сформированного стеганоключа показаны на рис. 5.15.

**Δ=**

| | 1 |
|---|---|
| 1 | -255 |
| 2 | -254 |
| 3 | -253 |
| 4 | -252 |
| ⋮ | ⋮ |
| 254 | -2 |
| 255 | -1 |
| 256 | 0 |
| 257 | 1 |
| 258 | 2 |
| ⋮ | ⋮ |
| 508 | 252 |
| 509 | 253 |
| 510 | 254 |
| 511 | 255 |

**b=**

| | 1 |
|---|---|
| 1 | 1 |
| 2 | 0 |
| 3 | 1 |
| 4 | 0 |
| ⋮ | ⋮ |
| 254 | 0 |
| 255 | 1 |
| 256 | 0 |
| 257 | 1 |
| 258 | 0 |
| ⋮ | ⋮ |
| 508 | 0 |
| 509 | 0 |
| 510 | 0 |
| 511 | 1 |

**Рис. 5.15. Фрагменты стеганоключа**

**Шаг 3**

Выполним развертывание массива контейнера **C** (массив синей цветовой составляющей) в вектор, используя модуль (М.16). Зададим стартовый индекс элемента полученного вектора, начиная с которого будет производиться встраивание бит сообщения (например, $\mu_S := 105$).

Для расчета величины шага (псевдослучайного интервала) используем модуль (М.15). Пусть при этом **K** := 8.

**Шаг 4**

Алгоритм встраивания реализует модуль (М.30). Формирование вектора двоичных данных из строки символов аналогично представленному в (М.21) (при этом, однако, **S** ← **C** необходимо заменить на **Sv** ← **Cv**).

Для каждого **μ**-го бита сообщения выполняется вычисление индекса **z** элемента вектора контейнера **Cv**. Рассчитывается разница **Δ'** между соседними пикселями $\mathbf{Cv_z}$ и $\mathbf{Cv_{z-1}}$. Внутренним циклом **i** ∈ 1..**rows**(Δ) производится поиск соответствующего значения разницы в векторе Δ. В случае обнаружения, переменной $\mathfrak{I}$ присваивается значение индекса **i**, который соответствует данной разнице в Δ.

Если значение $\mathbf{b}_{\mathfrak{I}}$ не соответствует текущему биту скрываемого сообщения, то выполняется поиск ближайшего индекса, при котором $\mathbf{b_i}$ равняется биту сообщения. Поиск производится вниз (**L**) и вверх (**H**) от индекса $\mathfrak{I}$.

Предварительное присвоение переменным $\mathfrak{I}_\mathbf{L}$ и $\mathfrak{I}_\mathbf{H}$ значения ±1000 обеспечивает невозможность дублирования предыдущих значений $\mathfrak{I}$, если движение вниз или вверх от $\mathfrak{I}$ не привело к выполнению поставленного условия (последнее возможно при нахождении индекса $\mathfrak{I}$ слишком близко к нижней или верхней границе вектора **b**). После того как значения $\mathfrak{I}_\mathbf{L}$ и $\mathfrak{I}_\mathbf{H}$ найдены, выбирается то из них, которое ближе к начальному значению $\mathfrak{I}$.

Интенсивность пикселя контейнера $\mathbf{Sv_z}$ равна увеличенной на величину $\Delta_{\mathfrak{I}_{\mathbf{L(H)}}}$ интенсивности смежного пикселя $\mathbf{Sv_{z-1}}$. Если данное увеличение приводит к выходу значения интенсивности цвета за пределы диапазона [0; 255], то, наоборот, интенсивности смежного пикселя $\mathbf{Sv_{z-1}}$ присваивается значение интенсивности пикселя $\mathbf{Sv_z}$ , уменьшенной на величину $\Delta_{\mathfrak{I}_{\mathbf{L(H)}}}$. После встраивания последнего бита сообщения внешний цикл прерывается.

Проводим обратное свертывание вектора **Sv** в матрицу, имеющую размерность первичного массива **C** (М.7). Получаем массив **S**.

```
Sv:= | ① -- см. (М.21)
     | z ← μ_s
     | for μ ∈ 1.. L_M
     |   | z ← z + step(D2B(z))
     |   | Δ' ← Cv_z − Cv_(z−1)
     |   | for i ∈ 1.. rows(Δ)
     |   |   if Δ' = Δ_i
     |   |     | ℑ ← i
     |   |     | break
     |   | if b_ℑ ≠ M_(vec_bin_μ)
     |   |   | for σ ∈ 1.. rows(Δ) − 1
     |   |   |   | ℑ_L ← −1000
     |   |   |   | if b_(ℑ−σ) = M_(vec_bin_μ)  if ℑ − σ ≥ 1
     |   |   |   |   | ℑ_L ← ℑ − σ
     |   |   |   |   | break
     |   |   | for σ ∈ 1.. rows(Δ) − 1
     |   |   |   | ℑ_H ← 1000
     |   |   |   | if b_(ℑ+σ) = M_(vec_bin_μ)  if ℑ + σ ≤ rows(Δ)
     |   |   |   |   | ℑ_H ← ℑ + σ
     |   |   |   |   | break
     |   |   | if ℑ − ℑ_L < ℑ_H − ℑ
     |   |   |   | Sv_z ← Sv_(z−1) + Δ_(ℑ_L)  if 0 ≤ Sv_(z−1) + Δ_(ℑ_L) ≤ 255
     |   |   |   | Sv_(z−1) ← Sv_z − Δ_(ℑ_L)  otherwise
     |   |   | if ℑ − ℑ_L ≥ ℑ_H − ℑ
     |   |   |   | Sv_z ← Sv_(z−1) + Δ_(ℑ_H)  if 0 ≤ Sv_(z−1) + Δ_(ℑ_H) ≤ 255
     |   |   |   | Sv_(z−1) ← Sv_z − Δ_(ℑ_H)  otherwise
     | Sv
```

(М.30)

**Шаг 5**

При извлечении сообщения предварительно формируется стеганоключ — векторы **Δ*** и **b***. Программные модули при этом идентичны (М.28) и (М.29). Массив контейнера **S*** развертывается в вектор **Sv*** (подобно (М.16)).

**Шаг 6**

Модуль извлечения (М.31) вычисляет разницу интенсивностей смежных пикселей $\mathbf{Sv}^*_{\mathbf{z}}$ и $\mathbf{Sv}^*_{\mathbf{z-1}}$ и выполняет поиск соответствующей разницы в кодовой таблице **Δ***.

Значение бита $\mathbf{b}^*_{\mathfrak{I}}$, соответствующее данной разнице, присваивается текущему элементу вектора **M***. В конце модуля вектор двоичных данных преобразуется в символьную строку.

Полученные при вычислении визуального искажения результаты сведены в табл. 5.1 (стр. 125).

```
M* := | z ← μ*_S
      | for μ ∈ 1 .. rows(Sv*)
      |   z ← z + step(D2B(z))
      |   Δ' ← Sv*_z − Sv*_(z−1)   if z ≤ rows(Sv*)
      |   break if z > rows(Sv*)
      |   for i ∈ 1 .. rows(Δ*)
      |     if Δ' = Δ*_i
      |       ℑ ← i
      |       break
      |   M*_μ ← b*_ℑ
      | for j ∈ 1 .. rows(M*) ÷ 8
      |   M* vec_j ← B2D(submatrix(M*, 8·j − 7, 8·j, 1, 1))
      | vec2str(M* vec)
```

(M.31)

#### 5.3.2.7. Метод Куттера-Джордана-Боссена

Куттер (M. Kutter), Джордан (F. Jordan) и Боссен (F. Bossen) [83] предложили алгоритм встраивания в канал синего цвета изображения, имеющего RGB-кодирование, поскольку к синему цвету ЗСЧ является наименее чувствительной. Рассмотрим алгоритм передачи одного бита секретной информации в предложенном методе.

Пусть $\boldsymbol{M_i}$ — бит, который подлежит встраиванию, $\boldsymbol{C} = \{\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}\}$ — изображение-контейнер, $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$ — псевдослучайный пиксель контейнера, в который будет выполняться встраивание.

Секретный бит $\boldsymbol{M_i}$ встраивается в канал синего цвета путем модификации яркости $\lambda_{x,y} = 0.29890 \cdot \boldsymbol{R}_{x,y} + 0.58662 \cdot \boldsymbol{G}_{x,y} + 0.11448 \cdot \boldsymbol{B}_{x,y}$ :

$$\boldsymbol{B}'_{x,y} = \begin{cases} \boldsymbol{B}_{x,y} - \upsilon \cdot \lambda_{x,y}, & \text{при } \boldsymbol{m}_i = 0; \\ \boldsymbol{B}_{x,y} + \upsilon \cdot \lambda_{x,y}, & \text{при } \boldsymbol{m}_i = 1. \end{cases} = \boldsymbol{B}_{x,y} + (2 \cdot \boldsymbol{m}_i - 1) \cdot \upsilon \cdot \lambda_{x,y}, \qquad (5.5)$$

где $\upsilon$ — константа, определяющая энергию встраиваемого сигнала. Ее величина зависит от назначения стеганосистемы. Чем больше $\upsilon$, тем выше устойчивость встроенной информации к искажениям, однако и тем сильнее ее заметность.

Получатель извлекает бит, не имея первичного изображения, то есть, «вслепую». Для этого выполняется предсказание значения первичного, не модифицированного пикселя на основе значений соседних пикселей. Для получения оценки пикселя предложено использовать значение нескольких пикселей, размещенных в том же столбце и в той же строке массива графического контейнера. Авторы использовали «крест» пикселей размером 7×7. Оценка $\hat{\boldsymbol{B}}^*_{x,y}$ получается в виде

$$\hat{\boldsymbol{B}}^*_{x,y} = \frac{1}{4 \cdot \sigma} \cdot \left[ \sum_{i=-\sigma}^{+\sigma} \boldsymbol{B}^*_{x+i,y} + \sum_{j=-\sigma}^{+\sigma} \boldsymbol{B}^*_{x,y+j} - 2 \cdot \boldsymbol{B}^*_{x,y} \right], \qquad (5.6)$$

где $\sigma$ — количество пикселей сверху (снизу, слева, справа) от оцениваемого пикселя (в случае креста 7×7 $\sigma = 3$).

При извлечении встроенного бита вычисляется разница $\delta$ между текущим ($\boldsymbol{B}^*_{x,y}$) и прогнозируемым ( $\hat{\boldsymbol{B}}^*_{x,y}$ ) значениями интенсивности пикселя $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$:

$$\delta = \boldsymbol{B}^*_{x,y} - \hat{\boldsymbol{B}}^*_{x,y}. \quad (5.7)$$

Знак $\delta$ будет означать встроенный бит: если $\delta < 0$, то $\boldsymbol{M}_i = 0$; если $\delta > 0$, то $\boldsymbol{M}_i = 1$.

Функции встраивания и извлечения в данном методе не симметричны, то есть, функция извлечения не является обратной функцией встраивания. Хотя, как указывают авторы метода, правильное распознание бита сообщения в случае применения описанных выше процедур является высоковероятным, однако не стопроцентным. Для уменьшения вероятности ошибок извлечения было предложено в процессе встраивания каждый бит повторять несколько раз (многократное встраивание). Поскольку при этом каждый бит был повторен $\tau$ раз, то получается $\tau$ оценок одного бита сообщения. Секретный бит извлекается по результатам усреднения разницы между реальным и оцененным значениями интенсивности пикселя в полученном контейнере:

$$\delta = \tau^{-1} \cdot \sum_{i=1}^{\tau} \left[ \boldsymbol{B}^*_{x,y} - \hat{\boldsymbol{B}}^*_{x,y} \right]. \quad (5.8)$$

Как и в предыдущем случае, знак усредненной разницы $\delta$ будет определять значение встроенного бита. В работе [83] показано, что алгоритм устойчив ко многим известным видам атак: НЧ фильтрации изображения, его компрессии в соответствии с алгоритмом JPEG, обрезанию краев.

Предлагается следующая реализация данного метода.

**Шаг 1**

Исходные данные — стандартные.

**Шаг 2**

Массив яркости получается с помощью функции **READBMP**("имя_файла"), возвращающей массив, который представляет изображение BMP-формата в яркостном формате (градациях серого): $\lambda$ := **READBMP**("C.bmp").

**Шаг 3**

Алгоритм встраивания реализуется программным модулем (М.32). При этом вычисление псевдослучайных координат пикселя, в который будет встраиваться бит сообщения, выполняется по алгоритму, описанному в подпункте 5.3.2.3 для метода псевдослучайной перестановки.

Предварительно задается параметр $\upsilon$, который определяет энергию встраиваемого сигнала, а также количество повторений встраивания одного и того же бита — $\tau$. Установлено, что результат встраивания визуально незаметен при значениях $\upsilon < 0{,}05$. Однако в этом случае для уменьшения ошибок при извлечении приходится значительно повышать количество повторных скрытий $\tau$ текущего бита ($\tau > 35$), что также негативно отражается на статистике изображения. Оптимальным, на наш взгляд, является значение $\upsilon \approx 0.15$ и $\tau < 20$, но, опять же, все зависит от характеристик изображения, которое было выбрано в качестве контейнера.

Вычисление индекса элемента двоичного вектора сообщения по формуле **ceil**($i/\tau$) позволяет один и тот же бит скрыть ровно $\tau$ раз, после чего будет встраиваться следующий бит сообщения и т.д.

$$
\begin{aligned}
&\mathbf{B'} := \Big| \ \textcircled{1} \ \text{-- см. (М.21), вместо } \mathbf{S} \leftarrow \mathbf{C} \text{ использовать } \mathbf{B'} \leftarrow \mathbf{B} \\
&\quad \text{for } \ i \in 1 .. \tau \cdot L_M \\
&\quad\quad x \leftarrow \text{floor}\left(\frac{i}{Y}\right) + 1 \\
&\quad\quad y \leftarrow \text{mod}(i, Y) + 1 \\
&\quad\quad \text{for } \ s \in 1 .. \Re \\
&\quad\quad\quad x \leftarrow \text{mod}\left(x + B2D\left(\overrightarrow{\left(D2B\left(K_{2 \cdot s-1}\right) \oplus \text{submatrix}(D2B(y), 1, 8, 1, 1)\right)}, X\right) + 1\right. \\
&\quad\quad\quad y \leftarrow \text{mod}\left(y + B2D\left(\overrightarrow{\left(D2B\left(K_{2 \cdot s}\right) \oplus \text{submatrix}(D2B(x), 1, 8, 1, 1)\right)}, Y\right) + 1\right. \\
&\quad\quad j \leftarrow \text{ceil}\left(\frac{i}{\tau}\right) \\
&\quad\quad B'_{x,y} \leftarrow B_{x,y} + \left(2 \cdot M^{\text{vec\_bin}_j} - 1\right) \cdot \upsilon \cdot \lambda_{x,y} \\
&\quad\quad B'_{x,y} \leftarrow 255 \ \text{ if } \ B'_{x,y} > 255 \\
&\quad\quad B'_{x,y} \leftarrow 0 \ \text{ if } \ B'_{x,y} < 0 \\
&\quad B'
\end{aligned}
\qquad \text{(М.32)}
$$

После модификации интенсивности пикселя, определенного координатами ($\mathbf{x}$, $\mathbf{y}$), выполняется коррекция значения результирующей интенсивности. Иначе, при начальном значении интенсивности цвета пикселя, например, 255, внесение бита сообщения «1» приведет не к возрастанию значения интенсивности, а, наоборот, — к его уменьшению в сторону темных оттенков. В другом случае, при очень низких начальных значениях интенсивности, в частности 0, внесение бита «0» может привести к получению отрицательного значения, которое будет восприниматься как значение интенсивности, близкое к 255.

**Шаг 4**

Перед извлечением сообщения должны быть известны:

- параметры контейнера;
- первичный ключ $\mathbf{K_o}^*$;
- количество циклов вычисления координат ($\mathbf{x}$, $\mathbf{y}$) $\Re^*$;
- количество дублирующих встраиваний одного бита $\tau^*$;
- размерность (конфигурация) креста $\sigma$ — количество пикселей сверху (снизу, слева, справа) от оцениваемого пикселя — рис. 5.16.

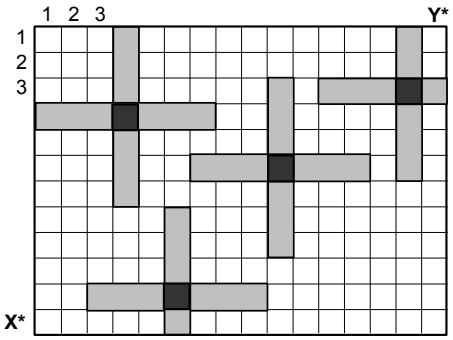



**Рис. 5.16. Примеры оцениваемых пикселей и оценивающих конфигураций («крестов»)**

Модуль извлечения скрытого сообщения (М.33) содержит в себе блок вычисления псевдослучайных координат (**x**, **y**), идентичный соответствующему блоку в модуле встраивания (М.32).

$$
\begin{aligned}
&\mathbf{M^*} := \left|\begin{array}{l}
\mathbf{t} \leftarrow 1 \\
\textbf{for}\ \ \mathbf{i} \in 1 .. \mathbf{X^*} \cdot \mathbf{Y^*} \\
\quad \mathbf{x} \leftarrow \mathbf{floor}(\mathbf{i} \div \mathbf{Y^*}) + 1 \\
\quad \mathbf{y} \leftarrow \mathbf{mod}(\mathbf{i}, \mathbf{Y^*}) + 1 \\
\quad \textbf{for}\ \ \mathbf{s} \in 1 .. \mathfrak{R}^* \\
\quad\quad \mathbf{x} \leftarrow \mathbf{mod}\left(\mathbf{x} + \mathbf{B2D}\left(\overrightarrow{\left(\mathbf{D2B}\left(\mathbf{K^*}_{2 \cdot s-1}\right) \oplus \mathbf{submatrix}(\mathbf{D2B}(\mathbf{y}), 1, 8, 1, 1)\right)}\right), \mathbf{X^*}\right) + 1 \\
\quad\quad \mathbf{y} \leftarrow \mathbf{mod}\left(\mathbf{y} + \mathbf{B2D}\left(\overrightarrow{\left(\mathbf{D2B}\left(\mathbf{K^*}_{2 \cdot s}\right) \oplus \mathbf{submatrix}(\mathbf{D2B}(\mathbf{x}), 1, 8, 1, 1)\right)}\right), \mathbf{Y^*}\right) + 1 \\
\quad \textbf{if}\ \ \sigma < \mathbf{x} \le \mathbf{X^*} - \sigma \\
\quad\quad \sigma 1 \leftarrow -\sigma \\
\quad\quad \sigma 2 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{x} \le \sigma \\
\quad\quad \sigma 1 \leftarrow 1 - \mathbf{x} \\
\quad\quad \sigma 2 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{x} > \mathbf{X^*} - \sigma \\
\quad\quad \sigma 1 \leftarrow -\sigma \\
\quad\quad \sigma 2 \leftarrow \mathbf{X^*} - \mathbf{x} \\
\quad \textbf{if}\ \ \sigma < \mathbf{y} \le \mathbf{Y^*} - \sigma \\
\quad\quad \sigma 3 \leftarrow -\sigma \\
\quad\quad \sigma 4 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{y} \le \sigma \\
\quad\quad \sigma 3 \leftarrow 1 - \mathbf{y} \\
\quad\quad \sigma 4 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{y} > \mathbf{Y^*} - \sigma \\
\quad\quad \sigma 3 \leftarrow -\sigma \\
\quad\quad \sigma 4 \leftarrow \mathbf{Y^*} - \mathbf{y} \\
\quad \mathbf{B^*\hat{}}_{\mathbf{t}} \leftarrow \dfrac{\sum_{a=\sigma 1}^{\sigma 2} \mathbf{B^*}_{\mathbf{x}+a, \mathbf{y}} + \sum_{a=\sigma 3}^{\sigma 4} \mathbf{B^*}_{\mathbf{x}, \mathbf{y}+a} - 2 \cdot \mathbf{B^*}_{\mathbf{x}, \mathbf{y}}}{\sigma 2 - \sigma 1 + \sigma 4 - \sigma 3} \\
\quad \mathbf{B^*}_{\mathbf{t}} \leftarrow \mathbf{B^*}_{\mathbf{x}, \mathbf{y}} \\
\quad \mathbf{t} \leftarrow \mathbf{t} + 1 \\
\quad \textbf{if}\ \ \mathbf{t} > \tau^* \\
\quad\quad \delta \leftarrow \dfrac{1}{\tau^*} \cdot \sum_{\tau=1}^{\tau^*} \left(\mathbf{B^*}_{\tau} - \mathbf{B^*\hat{}}_{\tau}\right) \\
\quad\quad \mathbf{j} \leftarrow \mathbf{ceil}(\mathbf{i} \div \tau^*) \\
\quad\quad \mathbf{M^*}\ \mathbf{vec\_bin}_{\mathbf{j}} \leftarrow 0 \quad \textbf{if}\ \ \delta \le 0 \\
\quad\quad \mathbf{M^*}\ \mathbf{vec\_bin}_{\mathbf{j}} \leftarrow 1 \quad \textbf{if}\ \ \delta > 0 \\
\quad\quad \mathbf{t} \leftarrow 1 \\
\textbf{for}\ \ \mathbf{j} \in 1 .. \mathbf{rows}\left(\mathbf{M^*}\ \mathbf{vec\_bin}\right) \div 8 \\
\quad \mathbf{M^*}\ \mathbf{vec}_{\mathbf{j}} \leftarrow \mathbf{B2D}\left(\mathbf{submatrix}\left(\mathbf{M^*}\ \mathbf{vec\_bin}, 8 \cdot \mathbf{j} - 7, 8 \cdot \mathbf{j}, 1, 1\right)\right) \\
\mathbf{vec2str}\left(\mathbf{M^*}\ \mathbf{vec}\right)
\end{array}\right. \qquad \text{(M.33)}
\end{aligned}
$$

Далее следуют блоки выполнения условий, которые в совокупности позволяют учесть проблемные случаи, когда оцениваемый пиксель находиться слишком близко к краю (краям) изображения и построить полноценный «крест» из окружающих пикселей не представляется возможным (см. рис. 5.16).

В модуле (M.33) предварительно проводится генерирование псевдослучайных индексов (**x**, **y**), определяющих элемент массива **B*** (пиксель), вокруг которого будет проводиться оценка близлежащих пикселей. По результатам генерирования данных индексов вычисляется количество пикселей сверху и снизу, слева и справа от оцениваемого. В дальнейшем производится вычисление оценки первичного значения оцениваемого пикселя (формула (5.6)). Полученный результат заносится в **t**-й элемент массива **B*^**. Оцениваемое значение пикселя $\mathbf{B}^*_{\mathbf{x,y}}$ сохраняется в буферном массиве $\mathbf{B}^*_{\mathbf{t}}$ (большая бета).

Если $\mathbf{t} \le \tau^*$, то продолжается накопление оценок пикселей, в которые был встроен один и тот же бит сообщения. В случае $\mathbf{t} > \tau^*$ итоги предыдущей оценки обобщаются: вычисляется усредненная разница $\delta$ между первичными и оцененными значениями интенсивностей пикселей, выступивших контейнерами для одного бита встраиваемых данных. В зависимости от знака полученной разницы, **j**-му элементу вектора двоичных данных (индекс элемента определяется по результату вычисления функции **ceil**(**i**/$\tau^*$)) присваивается значение 0 или 1. Переменная **t** сбрасывается в 1. Начинается сбор оценок значений интенсивности пикселей, в которые был встроен следующий бит сообщения. Процесс повторяется до тех пор, пока не будут проанализированы все элементы графического массива.

Результаты вычисления визуального искажения контейнера сведены в табл. 5.1.

#### 5.3.2.8. Метод Дармстедтера-Делейгла-Квисквотера-Макка

Нетрадиционный блочный метод встраивания в пространственную область контейнера предложили Дармстедтер (V. Darmstaedter), Делейгл (J.-F. Delaigle), Квисквотер (J.J. Quisquater) и Макк (B. Macq) [100]. Разработанный ими метод позволяет достичь компромисса между устойчивостью стеганосистемы к искажениям, качеством встраивания и, конечно же, вычислительной сложностью алгоритма. Метод базируется на элементарном перцепционном (ощущаемом) восприятии и позволяет приспосабливать встраивания относительно текущего содержимого блоков контейнера.

Перед встраиванием, конфиденциальная информация преобразуется в вектор двоичных данных. Каждый бит встраивается в отдельный блок. В рассмотренном авторами варианте размерность блоков составляла 8×8 пикселей. Главная причина такого выбора, очевидно, — соразмерность с блоками, которые используются при JPEG-компрессии. Таким образом, действие компрессии будет одинаково распространяться на каждый встроенный бит. Кроме того, при этом информация встраивается с избыточностью, что увеличивает общую устойчивость стеганосистемы.

В общем случае процесс встраивания бит сообщения выполняется в четыре этапа:

1. Разбиение массива изображения-контейнера на блоки 8×8 пикселей.
2. Классификация пикселей отдельного блока на *зоны* с приблизительно однородными значениями яркости.
3. Разбиение каждой зоны на *категории* в соответствии с индивидуальной (псевдослучайной) маской.

4. Встраивание бита в зависимости от соотношения между средними значениями категорий каждой зоны путем модификации значений яркости каждой категории в каждой зоне.

Рассмотрим последние три этапа более подробно.

#### Классификация на зоны

Цель состоит в том, чтобы разбить пиксели внутри блока на группы, которые имели бы приблизительно одинаковую яркость. Такая классификация принимает во внимание особенности блока, представляющие интерес с точки зрения невидимости и стойкости. При классификации авторы выделяют три типа контраста:

- ***резко выраженный контраст*** (рис. 5.17 *а*), когда можно различить две зоны, разделенные заметным скачком яркости;
- ***постепенный контраст*** (рис. 5.17 *б*), когда две однородные зоны разделены участком с постепенным изменением яркости;
- ***шумовой (нечеткий) контраст*** (рис. 5.17 *в*) с яркостью, распределенной наподобие случайного шума (в предельном случае шумовой контраст вырождается в однотонное изображение — контраст отсутствует, все пиксели блока имеют одинаковую яркость).

Отсортированные по возрастанию значения яркости пикселей блока можно представить возрастающей функцией $F(i)$, где $F(1)$ — наименьшее значение яркости среди всех присутствующих в данном блоке, а $F(N^2)$ — наибольшее среди присутствующих в блоке значений яркости ($N$ — размерность квадратного блока). Тип контраста блока определяет крутизна функции $F(i)$, которую обозначим через $S(i)$.

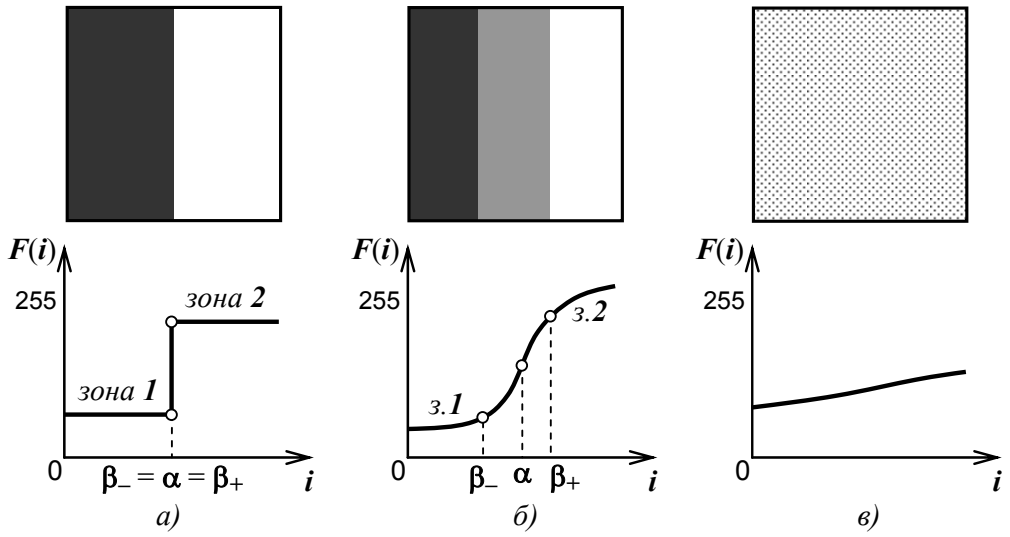



**Рис. 5.17. Классификация на зоны: *а* — резко выраженный контраст; *б* — постепенный контраст; *в* — шумовой контраст**

Пусть $S_{max}$ — максимальная крутизна функции $F$ при $i = \alpha$. Если $S_{max}$ ниже заданного порога $T_1$, считается, что блок имеет шумовой контраст. Если $S_{max}$ превышает порог $T_1$, блок имеет или постепенный, или резко выраженный контраст. В этом случае дополнительно определяют параметры $\beta_+$ и $\beta_-$ — индексы в ближайшей

окрестности точки $\alpha$ (соответственно выше и ниже ее), которые удовлетворяют неравенствам

$$S(\alpha) - S(\beta_+) > T_2 \quad \text{и} \quad S(\alpha) - S(\beta_-) > T_2, \tag{5.9}$$

где $T_2$ — еще одно заданное значение порога.

Если контраст резко выражен, то $\beta_+ \approx \alpha$ и $\beta_- \approx \alpha$. Если контраст постепенный, то интервал $[\beta_+, \beta_-]$ представляет собой переходную зону постепенного контраста.

Классификация пикселей $p(x, y)$ на две зоны определяется следующими правилами:

- для постепенного и резко выраженного контрастов:
  - если $p(x, y) \le F(\beta_-)$, то пиксель $p(x, y)$ принадлежит к зоне 1;
  - если $p(x, y) \ge F(\beta_+)$, то пиксель $p(x, y)$ принадлежит к зоне 2;
  - если $F(\beta_-) < p(x, y) < F(\beta_+)$, то пиксель $p(x, y)$ принадлежит к переходной зоне.
- для шумового контраста пиксели распределяют на две зоны одинаковой размерности:
  - если $p(x, y) < F(N^2/2)$, то пиксель $p(x, y)$ принадлежит к зоне 1;
  - если $p(x, y) > F(N^2/2)$, то пиксель $p(x, y)$ принадлежит к зоне 2.

В блоках первого и второго типов зоны с разной яркостью не обязательно должны размещаться вплотную друг к другу и не обязательно должны содержать равное количество пикселей. Более того, некоторые пиксели вообще могут не принадлежать ни к одной из этих зон. В блоках третьего типа классификация более сложная.

#### Разбиение зон на категории

После разбиения на зоны необходимо предусмотреть встраивание бита путем модификации определенных характеристик зон. К сожалению, как указывают авторы, непосредственное влияние на зоны приводит к результатам, которые или недостаточно устойчивы к помехам, или же являются неудовлетворительными исходя из показателей визуальных искажений исходного изображения.

Поиск оптимального для встраивания пикселя заключается в разделении зоны на две категории ($A$ и $Z$). Для сортировки пикселей по этим категориям на блоки изображения накладываются маски, причем желательна индивидуальность масок для каждого конкретного блока. Предназначение масок заключается в обеспечении секретности встраивания.

Примеры масок для двух зон представлены на рис. 5.18.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | A | A | A | Z | Z | Z | Z |
| A | A | A | A | Z | Z | Z | Z |
| A | A | A | A | Z | Z | Z | Z |
| A | A | A | A | Z | Z | Z | Z |
| Z | Z | Z | Z | A | A | A | A |
| Z | Z | Z | Z | A | A | A | A |
| Z | Z | Z | Z | A | A | A | A |
| Z | Z | Z | Z | A | A | A | A |

*а)*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | A | Z | Z | A | A | Z | Z |
| A | A | Z | Z | A | A | Z | Z |
| Z | Z | A | A | Z | Z | A | A |
| Z | Z | A | A | Z | Z | A | A |
| A | A | Z | Z | A | A | Z | Z |
| A | A | Z | Z | A | A | Z | Z |
| Z | Z | A | A | Z | Z | A | A |
| Z | Z | A | A | Z | Z | A | A |

*б)*

**Рис. 5.18. Примеры используемых масок: *а* — размерами 4×4; *б* — размерами 2×2**

Авторы рекомендуют использовать более сложные комбинации и изменять маску при переходе к скрытию каждого следующего бита сообщения. Категория, к которой будет отнесен тот или иной пиксель, зависит от двух факторов:

- пространственного размещения пикселя в массиве блока;
- номера зоны, к которой был отнесен пиксель.

Важно отметить, что алгоритм формирования масок должен держаться в секрете, поскольку знание конфигурации последних существенно снижает устойчивость стеганосистемы в целом.

#### Правила встраивания бит сообщения

По результатам выполнения первых трех этапов получены четыре разные группы пикселей в определенных блоках: в зависимости от зоны (1 или 2) и категорий (***A*** и ***Z***). Следует отметить, что существует еще и пятая группа пикселей: те, которые не вошли ни в одну из зон. Однако последние не принимают участия в дальнейшем анализе.

Для указанных четырех подмножеств могут быть вычислены шесть параметров:

- четыре средних значения яркости $\lambda_{1A}$, $\lambda_{1Z}$, $\lambda_{2A}$ и $\lambda_{2Z}$ для групп, которые содержат соответственно $n_{1A}$, $n_{1Z}$, $n_{2A}$ и $n_{2Z}$ пикселей;
- два средних значения яркости соответствующих зон: $\Lambda_1$ и $\Lambda_2$.

Средние значения яркости одинаковых зон объединяются вместе. Таким образом, один бит сообщения встраивается в каждую из зон. Это увеличивает устойчивость системы и позволяет встраивать бит без чрезмерного искажения блока.

Встраивание бита ***b*** в блок выполняется в соответствии со связями между категориями средних значений яркости. Правило встраивания следующее:

$$\begin{cases} \lambda^*_{1Z} - \lambda^*_{1A} = E; \\ \lambda^*_{2Z} - \lambda^*_{2A} = E, \end{cases} \text{ при } b = 0; \qquad (5.10\ a)$$

$$\begin{cases} \lambda^*_{1A} - \lambda^*_{1Z} = E; \\ \lambda^*_{2A} - \lambda^*_{2Z} = E, \end{cases} \text{ при } b = 1, \qquad (5.10\ б)$$

где $\lambda^*_{1A}$, $\lambda^*_{1Z}$, $\lambda^*_{2A}$ и $\lambda^*_{2Z}$ — средние значения яркости, необходимые для скрытия бита ***b***; ***E*** — уровень встраивания, то есть, необходимая разница между указанными средними значениями.

Для того чтобы сделать результат встраивания как можно более незаметным, должны быть сохранены низкие частоты (к которым, напомним, наиболее чувствительна ЗСЧ). Сохранение средних значений интенсивностей каждой зоны обеспечивается выполнением следующих условий:

$$\frac{n_{1A} \cdot \lambda^*_{1A} + n_{1Z} \cdot \lambda^*_{1Z}}{n_{1A} + n_{1B}} = \Lambda_1; \qquad (5.11\ a)$$

$$\frac{n_{2A} \cdot \lambda^*_{2A} + n_{2Z} \cdot \lambda^*_{2Z}}{n_{2A} + n_{2B}} = \Lambda_2. \qquad (5.11\ б)$$

Формулы (5.10) и (5.11) позволяют определить значения $\lambda^*_{1A}$, $\lambda^*_{1Z}$, $\lambda^*_{2A}$ и $\lambda^*_{2Z}$. Яркость пикселей каждой зоны должна быть адаптирована для сохранений значений $\Lambda_1$ и $\Lambda_2$. При этом считается, что изменение яркости всех пикселей, которые принадлежат к одной зоне, одинаковое.

Пусть $\Delta_{1A}$, $\Delta_{1Z}$, $\Delta_{2A}$ и $\Delta_{2Z}$ — изменение яркости. Тогда имеем:

$$\Delta_{ij} = \lambda^{*}_{ij} - \lambda_{ij}, \tag{5.12}$$

где $i = \{1, 2\}$; $j = \{A, Z\}$.

#### Извлечение встроенной информации

Извлечение встроенной информации из контейнера требует наличия сведений о размерности блоков, на которые разбивается изображение, а также о конфигурации масок, которые использовались при встраивании. Процесс извлечения состоит из следующих этапов:

1. Разбиение изображения на блоки размерностью $N \times N$.
2. Классификация пикселей отдельного блока на зоны.
3. Деление каждой зоны на категории.
4. Сопоставление средних значений яркости для определения значения встроенного бита данных.

Первые три этапа идентичны соответствующим этапам алгоритма встраивания. Четвёртый этап заслуживает более подробного рассмотрения.

Пусть $\Sigma_1$ и $\Sigma_2$ — значения, полученные путем сравнения средних значений яркости:

$$\Sigma_1 = \lambda_{1A} - \lambda_{1B}; \tag{5.13 а}$$

$$\Sigma_2 = \lambda_{2A} - \lambda_{2B}. \tag{5.13 б}$$

Знак вычисленных $\Sigma_1$ и $\Sigma_2$ позволяет сделать предположение относительно истинного значения скрытого бита. Кроме того, абсолютные значения $\Sigma_1$ и $\Sigma_2$ несут информацию об уровне достоверности такого предположения.

Возможны три случая:

***Случай 1:*** $\Sigma_1 \cdot \Sigma_2 > 0$. При этом $b^* = 1$, если $\Sigma_1 > 0$, и $b^* = 0$, если $\Sigma_2 < 0$.

Уровень достоверности:

- очень высокий, если $|\Sigma_1|$ и $|\Sigma_2| > 2$;
- очень высокий, если $|\Sigma_1|$ или $|\Sigma_2| > 2.5$;
- низкий, если $|\Sigma_1|$ и $|\Sigma_2| < 0.7$;
- высокий во всех остальных случаях.

***Случай 2:*** $\Sigma_1 \cdot \Sigma_2 < 0$. Дополнительно вычисляется следующий параметр:

$$\Sigma' = \Sigma_1 \cdot (n_{1A} + n_{1Z}) + \Sigma_2 \cdot (n_{2A} + n_{2Z}).$$

При этом $b^* = 1$, если $\Sigma' > 0$, и $b^* = 0$, если $\Sigma' < 0$. Уровень достоверности при этом низкий.

***Случай 3:*** $\Sigma_1 \cdot \Sigma_2 \approx 0$. Вычисляется параметр $\Sigma' = \mathbf{max}(|\Sigma_1|, |\Sigma_2|)$. При этом $b^* = 1$, если $\Sigma' > 0$, и $b^* = 0$, если $\Sigma' < 0$. Уровень достоверности при этом низкий. Если $\Sigma' = 0$, то значение бита неопределенное.

Для повышения помехоустойчивости авторы предлагают использовать циклический код коррекции ошибок БЧХ.

Рассмотрим реализацию описанного выше метода в программной среде MathCAD.

**Шаг 1**

Выделим массивы цветовых компонентов изображения:

**R** := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

Аналогично предыдущим методам, встраивание будем выполнять в массив синей цветовой составляющей (**B**).

**Шаг 2**

Определяем размерность массива контейнера **X** и **Y**, а также задаем размерность сегментов (блоков), на которые он будет разбиваться: **X** := **rows**(**B**), **X** = 128; **Y** := **cols**(**B**), **Y** = 128; **N** := 8. При этом общее количество сегментов $\mathbf{N_S} := \mathbf{X} \cdot \mathbf{Y} / \mathbf{N}^2$, $\mathbf{N_S}$ = 256 сегментов.

**Шаг 3**

С помощью программного модуля (М.34) разбиваем общий массив **B** на отдельные блоки. При этом последние выделяются из массива с помощью функции **submatrix**(•) и сохраняются как элементы вектора **S**.

```
S := | c1 ← 1
     | c2 ← N
     | for b ∈ 1 .. N_S
     |     | r1 ← mod[N·(b − 1) + 1 , X]
     |     | r2 ← r1 + N − 1
     |     | S_b ← submatrix(B , r1 , r2 , c1 , c2)
     |     | if r2 = X
     |     |     | c1 ← c1 + N
     |     |     | c2 ← c2 + N
     | S
```

(М.34)

На начальном этапе, столбцы, которые ограничивают выделенный из массива сегмент, имеют индексы **c1** ← 1, **c2** ← **N**. Участок, ограниченный столбцами **c1** и **c2**, полностью проходится путем постепенного наращивания значения индексов строк **r1** и **r2**. В случае, когда индекс **r2** совпадает с индексом последней строки, индексы столбцов увеличиваются на ширину выделяемых сегментов **N** и т.д.

Отметим, что данный модуль не рассчитан на разбиение изображения на сегменты, размерность которых не кратна размерности изображения (данную возможность можно учесть путем введения в тело программного модуля соответствующих ограничивающих условий). В данном же случае, в результате вычисления программного модуля возвращается вектор **S** из 256 элементов, каждый из которых представляет собой матрицу размерностью 8×8.

Каждый сегмент **S** предназначен для скрытия одного бита секретного сообщения **M**, поэтому желательно предварительно проверить достаточность количества сегментов для этой операции. В нашем случае сообщение идентично используемому при рассмотрении предыдущих методов; его длина $\mathbf{L_M}$ = 200 бит. Таким образом, количество сегментов полностью достаточно, поскольку $\mathbf{N_S}$ = 256.

**Шаг 4**

Выполним классификацию пикселей каждого из блоков на зоны 1 и 2 (М.35). Во внимание принимается только количество сегментов, достаточное для встраивания данного сообщения **M**.

```
Zone := | for s ∈ 1 .. L_M
        |   Б ← S_s
        |   f ← Б^⟨1⟩
        |   for k ∈ 2 .. N
        |     f ← stack(f, Б^⟨k⟩)
        |   F ← sort(f)
        |   r ← 10
        |   χ_1 ← 1
        |   χ_r ← N^2
        |   for x ∈ 2 .. r − 1
        |     χ_x ← (x − 1)·round[N^2 ÷ (r − 1)]
        |   for x ∈ 1 .. rows(χ)
        |     ϕ_x ← F_(χ_x)
        |   spline ← lspline(χ, ϕ)
        |   S_max ← 0
        |   α ← 0
        |   for ω ∈ 1 .. N^2
        |     if d/dω interp(spline, χ, ϕ, ω) > S_max
        |       | S_max ← d/dω interp(spline, χ, ϕ, ω)
        |       | α ← ω
        |   α ← N^2/2  if α = 0
        |   α ← 2  if α = 1
        |   α ← (N^2 − 1)  if α = N^2
        |   for i ∈ 1 .. N                    if S_max < T_1
        |     for j ∈ 1 .. N
        |       Zone'_(i,j) ← mod(j + i, 2) + 1
        |   if S_max ≥ T_1
        |     for x ∈ α .. 1
        |       | β_ ← α  if x = 1
        |       | if (F_α − F_x) > T_2
        |       |   | β_ ← x
        |       |   | break
        |     for x ∈ α .. N^2
        |       | β_+ ← α  if x = N^2
        |       | if (F_x − F_α) > T_2
        |       |   | β_+ ← x
        |       |   | break
        |     for i ∈ 1 .. N
        |       for j ∈ 1 .. N
        |         | Zone'_(i,j) ← 1  if Б_(i,j) ≤ F_β_
        |         | Zone'_(i,j) ← 2  if Б_(i,j) ≥ F_β_+
        |         | Zone'_(i,i) ← "--" if F_β_ < Б_(i,j) < F_β_+
        |   Zone_s ← Zone'
        | Zone
```

(M.35)

В начале цикла изменения **s** переменной **Б** присваивается значение **s**-го сегмента **S**. Полученный массив **Б** разворачивается в вектор **f**, который в отсортированном виде присваивается переменной **F**. Таким образом, вектор **F** имеет $N^2$ элементов, расположенных по возрастанию.

Для существования возможности принятия одинаковых решений как на передающей стороне при встраивании, так и на принимающей стороне при извлечении относительно точки, при которой функция **F** имеет наибольшую крутизну (см. рис. 5.17), последнюю желательно сгладить, используя в качестве узловых точек лишь некоторую часть (**r**) от общего их количества ($N^2$). Очевидно, что абсцисса первой узловой точки должна равняться единице ($\chi_1 \leftarrow 1$), а последней — $\chi_r \leftarrow N^2$. Абсциссы промежуточных узловых точек формируются по выражению, смысл которого очевиден. В массив $\phi$ заносятся соответствующие ординаты узловых точек.

Пользуясь встроенной функцией **lspline**(•), формируем вектор кубических сплайн-коэффициентов вторых производных при приближении к опорным точкам (при этом предельные точки линейные). Для проведения сплайн-аппроксимации используем функцию **interp**(**spline**, $\chi$, $\phi$, $\omega$), которая для каждой искомой абсциссы $\omega$ вычисляет значение аппроксимированной функции.

Из основ дифференциального исчисления известно [101], что угол наклона касательной в заданной точке к графику функции равен арктангенсу от производной этой функции при данном значении аргумента. Следовательно, абсциссу точки $\alpha$ максимальной крутизны сплайн-аппроксимированной функции $\phi$ можно найти по максимуму первой производной ($d/d\omega$) данной функции среди всех возможных значений $\omega$.

В случае нахождения максимального значения крутизны, переменной $\alpha$ присваивается соответствующее значение $\omega$. Если для любого $\omega$ не найдено производной больше $S_{max}$ (все пиксели сегмента имеют одинаковую интенсивность), то переменной $\alpha$ присваивается значение $N^2/2$. Если точка максимальной крутизны является предельной (то есть, равна 1 или $N^2$), то она сдвигается на один шаг вправо или влево соответственно (при отсутствии такой поправки будут отсутствовать пиксели, которые принадлежат соответственно к первой или второй зоне).

В том случае, если полученное значение крутизны будет меньше порога ($T_1$), пиксели делятся между двумя зонами 1 и 2 поровну в шахматном порядке. Если $S_{max} \geq T_1$, выполняется поиск абсцисс $\beta_-$ и $\beta_+$. При значении интенсивности пикселя $Б_{i,j} \leq F_{\beta_-}$ последний относится к зоне 1; при $Б_{i,j} \geq F_{\beta_+}$ — к зоне 2; при $F_{\beta_-} < Б_{i,j} < F_{\beta_+}$ — к переходной зоне ("--").

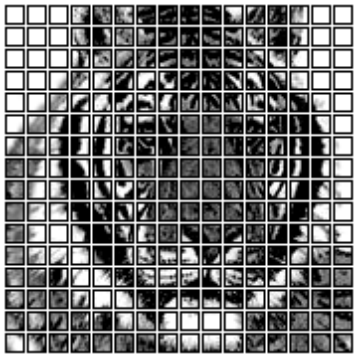
В результате вычисления модуля возвращается вектор **Zone**, каждый элемент которого представляет собой матрицу идентичной размерности с соответствующим сегментом изображения ($N \times N$). В свою очередь, каждый элемент такой матрицы может принимать три значения: 1, 2 или "--", что определяет, к какой из возможных зон принадлежит пиксель сегмента **S** с соответствующими координатами.

Представим результаты разбиения изображения на сегменты и классификации пикселей по зонам для некоторых из этих сегментов (рис. 5.19).

Нумерация сегментов следующая: в первом столбце — с 1-го по 16-й, во втором — с 17-го по 32-й и т.д. Предварительно установлены следующие значения порогов: $T_1 := 6$, $T_2 := 3$.

**Шаг 5**

Разбиваем каждую зону на категории "**A**" и "**Z**" в соответствии с индивидуальной маской. Простой программный модуль генерирования псевдослучайных масок изображено ниже (М.36).



$Zone_3$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |

$Zone_7$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | --" | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| --" | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$Zone_{40}$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | --" | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |

*а)* *б)* *в)* *г)*

**Рис. 5.19. Схема разбиения изображения на сегменты (*а*) и примеры классификации пикселей 3-го (*б*), 7-го (*в*) и 40-го (*г*) сегментов по зонам**

$$\mu := \left| \begin{array}{l} \text{for } i \in 1..N_S \\ \quad \left| \begin{array}{l} \text{for } j \in 1..N^2 \\ \quad \mu'_j \leftarrow 0 \\ \text{for } j \in 1..N^2 \div 2 \\ \quad \mu'_{\text{mod}\left(i+j\cdot K_0, N^2\right)+1} \leftarrow \text{"A"} \\ \text{for } j \in 1..N^2 \\ \quad \mu'_j \leftarrow \text{"Z"} \quad \text{if } \mu'_j = 0 \\ \mu_i \leftarrow \text{submatrix}(\mu', 1, N, 1, 1) \\ \text{for } c \in 2..N \\ \quad \mu_i \leftarrow \text{augment}\left[\mu_i, \text{submatrix}\left[\mu', c\cdot N-(N-1), c\cdot N, 1, 1\right]\right] \end{array} \right. \\ \mu \end{array} \right. \qquad \text{(M.36)}$$

Для каждого из $N_S$ сегментов изображения генерируется массив $\mu$ размерностью $N \times N$. В начале такого генерирования создается вектор $\mu'$ нулевых элементов размерностью $N^2$. Половине этих элементов присваивается символьное значение "A", причем это происходит с псевдослучайным шагом, который определяется заданным значением параметра $K_0$ и текущими значениями переменных цикла **i** и **j** (для того чтобы не выйти за рамки размерности вектора, используется функция возвращения остатка от деления). Всем другим элементам присваивается значение "Z". После заполнения, вектор $\mu'$ сворачивается в матрицу $\mu$.

Примеры некоторых масок, полученных для $K_0 := 123$, изображены на рис. 5.20.

$\mu_1$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |
| "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" | "Z" |
| "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" |
| "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" |
| "A" | "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" |
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |
| "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" | "Z" |
| "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" |

$\mu_2$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| "A" | "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" |
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |
| "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" | "Z" |
| "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" |
| "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" |
| "A" | "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" |
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |
| "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" | "Z" |

$\mu_3$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" |
| "A" | "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" |
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |
| "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" | "Z" |
| "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" |
| "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" | "A" |
| "A" | "Z" | "A" | "Z" | "Z" | "A" | "Z" | "A" |
| "A" | "Z" | "Z" | "A" | "Z" | "A" | "A" | "Z" |

**Рис. 5.20. Примеры масок разбиения по категориям для сегментов 1–3**

**Шаг 6**

Модуль встраивания бит сообщения (М.37) построен на основе выражений (5.10)–(5.12). В начале модуля создается вектор двоичных данных на основе скрываемого сообщения **M**. Циклом изменения **s** выполняется встраивание отдельного бита $\mathbf{b} \leftarrow \mathbf{M}_{\mathbf{vec\_bin_s}}$ в блок $\mathbf{Б} \leftarrow \mathbf{S_s}$ с учетом соответствующего массива зон $\mathbf{Z} \leftarrow \mathbf{Zone_s}$ и маски $\mathbf{м} \leftarrow \mu_{\mathbf{s}}$.

$$
\begin{array}{l}
\Delta := \left| \begin{array}{l}
\textcircled{I} \text{ -- см. (М.21), кроме } \mathbf{s} \leftarrow \mathbf{C} \\
\textbf{for } \mathbf{s} \in 1..\mathbf{L_M} \\
\quad \left| \begin{array}{l}
\mathbf{b} \leftarrow \mathbf{M}_{\mathbf{vec\_bin_s}} \\
\mathbf{Б} \leftarrow \mathbf{S_s} \\
\mathbf{Z} \leftarrow \mathbf{Zone_s} \\
\mathbf{м} \leftarrow \mu_{\mathbf{s}} \\
\mathbf{n} \leftarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\
\textbf{for } \mathbf{i} \in 1..\mathbf{N} \\
\quad \textbf{for } \mathbf{j} \in 1..\mathbf{N} \\
\quad\quad \left| \begin{array}{l}
\mathbf{n}_{1,1} \leftarrow \mathbf{n}_{1,1} + 1 \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 1 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"A"} \\
\mathbf{n}_{1,2} \leftarrow \mathbf{n}_{1,2} + 1 \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 1 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"Z"} \\
\mathbf{n}_{2,1} \leftarrow \mathbf{n}_{2,1} + 1 \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 2 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"A"} \\
\mathbf{n}_{2,2} \leftarrow \mathbf{n}_{2,2} + 1 \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 2 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"Z"}
\end{array} \right. \textcircled{а} \\
\Sigma\lambda \leftarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\
\textbf{for } \mathbf{i} \in 1..\mathbf{N} \\
\quad \textbf{for } \mathbf{j} \in 1..\mathbf{N} \\
\quad\quad \left| \begin{array}{l}
\Sigma\lambda_{1,1} \leftarrow \Sigma\lambda_{1,1} + \mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 1 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"A"} \\
\Sigma\lambda_{1,2} \leftarrow \Sigma\lambda_{1,2} + \mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 1 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"Z"} \\
\Sigma\lambda_{2,1} \leftarrow \Sigma\lambda_{2,1} + \mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 2 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"A"} \\
\Sigma\lambda_{2,2} \leftarrow \Sigma\lambda_{2,2} + \mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \textbf{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}} = 2 \wedge \mathbf{м}_{\mathbf{i},\mathbf{j}} = \text{"Z"}
\end{array} \right. \textcircled{б} \\
\overrightarrow{\lambda \leftarrow \left( \frac{\Sigma\lambda}{\mathbf{n}} \right)} \\
\textbf{for } \mathbf{x} \in 1..2 \\
\quad \left| \begin{array}{l}
\Lambda_{\mathbf{x}} \leftarrow \dfrac{\lambda_{\mathbf{x},1} \cdot \mathbf{n}_{\mathbf{x},1} + \lambda_{\mathbf{x},2} \cdot \mathbf{n}_{\mathbf{x},2}}{\mathbf{n}_{\mathbf{x},1} + \mathbf{n}_{\mathbf{x},2}} \\
\lambda'_{\mathbf{x}} \leftarrow \mathbf{lsolve}\left[ \begin{pmatrix} \mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2} \\ 1 & -1 \end{pmatrix}, \begin{bmatrix} \Lambda_{\mathbf{x}} \cdot (\mathbf{n}_{\mathbf{x},1} + \mathbf{n}_{\mathbf{x},2}) \\ \mathbf{E} \end{bmatrix} \right] \quad \textbf{if} \quad \mathbf{b} = 1 \\
\lambda'_{\mathbf{x}} \leftarrow \mathbf{lsolve}\left[ \begin{pmatrix} \mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2} \\ -1 & 1 \end{pmatrix}, \begin{bmatrix} \Lambda_{\mathbf{x}} \cdot (\mathbf{n}_{\mathbf{x},1} + \mathbf{n}_{\mathbf{x},2}) \\ \mathbf{E} \end{bmatrix} \right] \quad \textbf{if} \quad \mathbf{b} = 0
\end{array} \right. \\
\lambda^* \leftarrow \mathbf{augment}(\lambda'_1, \lambda'_2)^{\mathbf{T}} \\
\Delta_{\mathbf{S}} \leftarrow \lambda^* - \lambda
\end{array} \right. \\
\Delta
\end{array}
\qquad \text{(М.37)}
$$

Рассмотрим процедуру встраивания подробнее. На основе данных о принадлежности пикселей **s**-го блока к зоне 1 или 2, а также к категории "A" или "Z", подсчитывается количество пикселей, удовлетворяющих одной из возможных комбинаций зон и категорий (1A, 1Z, 2A, 2Z). Результаты заносятся в соответствующий элемент массива **n** (рис. 5.21), который в начале подсчета имел нулевое значение.


**Рис. 5.21. Конфигурация массивов n и Σλ**

Аналогично подсчитывается общая яркость $\Sigma\lambda$ пикселей, принадлежащих той или иной паре зон и категорий (конфигурация заполненного при этом массива соответствует предыдущему случаю — рис. 5.21).

По полученным результатам вычисляется массив $\lambda$ средних значений яркости каждой из четырех групп пикселей. Для сокращения программных строк данное вычисление происходит с использованием операции векторизации — одновременного выполнения некоторой скалярной математической операции (в нашем случае — деления) над всеми элементами массива(-ов), помеченными знаком векторизации «$\vec{}$». Конечно, такая параллельность вычисления относится не к самим вычислениям, а только к их алгоритмической записи. Поэтому кардинального изменения времени выполнения операции ждать при этом не приходится.

Далее для каждой зоны рассчитываются средние значения яркости $\Lambda$ (формулы (5.11)), которые необходимо сохранить и после встраивания бита в сегмент изображения. Для решения системы уравнений (5.10), (5.11) с двумя неизвестными ($\lambda^*_{1A}$ и $\lambda^*_{1Z}$ или $\lambda^*_{2A}$ и $\lambda^*_{2Z}$) используется встроенная функция MathCAD решения линейной системы из ***n*** уравнений при ***n*** неизвестных — **lsolve**(*H*,*V*), где *H* — квадратная несингулярная матрица; *V* — вектор, который имеет то же количество строк, что и матрица *H*.

Для удобства изложения, перепишем данные системы в соответствии с принятыми в программном модуле (М.37) обозначениями:

- при **b** = 1:

$$\left.\begin{aligned} \mathbf{n}_{\mathbf{x},1}\cdot\lambda_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\cdot\lambda_{\mathbf{x},2}=\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right),\\ \lambda_{\mathbf{x},1}-\lambda_{\mathbf{x},2}=\mathbf{E}; \end{aligned}\right\} \Rightarrow H=\begin{pmatrix} \mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ 1 & -1 \end{pmatrix},\ V=\begin{pmatrix} \Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E} \end{pmatrix};$$

- при **b** = 0:

$$\left.\begin{aligned} \mathbf{n}_{\mathbf{x},1}\cdot\lambda_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\cdot\lambda_{\mathbf{x},2}=\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right),\\ \lambda_{\mathbf{x},2}-\lambda_{\mathbf{x},1}=\mathbf{E}; \end{aligned}\right\} \Rightarrow H=\begin{pmatrix} \mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ -1 & 1 \end{pmatrix},\ V=\begin{pmatrix} \Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E} \end{pmatrix},$$

где **x** — номер зоны.

При любом значении бита **b**, функция **lsolve**(•) вычисляется дважды (для каждой из зон). Результатом ее вычисления является вектор из двух элементов, значения которых соответствуют искомым неизвестным. Полученные два вектора объединяем столбец к столбцу, а результат объединения транспонируем (чтобы перейти к конфигурации, изображенной на рис. 5.21).

На заключительном этапе вычисляется разница (5.12). Результат вычисления (который, разумеется, также имеет конфигурацию, соответствующую рис. 5.21) присваивается **s**-му элементу вектора $\Delta$.

**Шаг 7**

Модифицируем блоки изображения, используя программный модуль (М.38). При этом, если номер блока **s** превышает общее количество бит в сообщении ($L_M$), то модификация блока не проводится.

```
SM := | for s ∈ 1 .. N_S
      |   Б ← S_s
      |   if s ≤ L_M
      |     Б' ← Б
      |     ∇ ← Δ_S
      |     Z ← Zone_S
      |     м ← μ_S
      |     for i ∈ 1 .. N
      |       for j ∈ 1 .. N
      |         Б'_i,j ← Б_i,j + ∇_1,1  if Z_i,j = 1 ∧ м_i,j = "A"
      |         Б'_i,j ← Б_i,j + ∇_1,2  if Z_i,j = 1 ∧ м_i,j = "Z"
      |         Б'_i,j ← Б_i,j + ∇_2,1  if Z_i,j = 2 ∧ м_i,j = "A"
      |         Б'_i,j ← Б_i,j + ∇_2,2  if Z_i,j = 2 ∧ м_i,j = "Z"
      |   Б' ← Б  if s > L_M
      |   SM_s ← Б'
      | SM
```

(М.38)

**Шаг 8**

Возвращаем блоки на соответствующие им места в изображении, используя модуль (М.39). На первом этапе первые **X/N** блоков (в нашем случае — 16) объединяются в один общий массив путем объединения последней строки предыдущего блока с первой строкой последующего (функция **stack(•)**). В дальнейшем с помощью цикла подобная операция повторятся над каждой следующей группой из **X/N** блоков. Сгруппированные таким образом блоки параллельно объединяются между собой столбец к столбцу (функция **augment(•)**).

```
BM := | BM ← SM_1
      | for b ∈ 2 .. X÷N
      |   BM ← stack(BM , SM_b)
      | BM' ← 0
      | for b ∈ X÷N + 1 .. N_S
      |   BM' ← SM_b  if BM' = 0
      |   BM' ← stack(BM' , SM_b)  otherwise
      |   if mod(b , X÷N) = 0
      |     BM ← augment(BM , BM')
      |     BM' ← 0
      | BM
```

(М.39)

Воссозданное на основе массива **BM** изображение в большинстве случаев будет слишком искаженным (рис. 5.22) из-за уже упомянутого выше выхода значений интенсивностей пикселей за пределы диапазона [0; 255]. Поэтому данный массив необходимо дополнительно пронормировать, для чего, например, можно воспользоваться программным модулем (М.40).

$$
\begin{aligned}
&BM_{norm} := \left|
\begin{array}{l}
BM_{min} \leftarrow \min(BM) \\
BM_{max} \leftarrow \max(BM) \\
\text{for } x \in 1..X \\
\quad \text{for } y \in 1..Y \\
\qquad BM\,norm_{x,y} \leftarrow \text{round}\left( \dfrac{BM_{x,y} + |BM_{min}|}{BM_{max} + |BM_{min}|} \cdot 255 \right) \\
BM_{norm}
\end{array}
\right.
\end{aligned}
\qquad \text{(М.40)}
$$

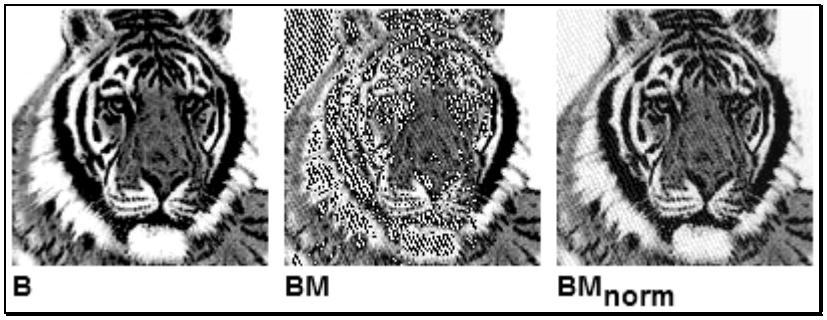



**Рис. 5.22. Сравнение первичного (B), модифицированного (BM) (при значении E = 35) и дополнительного нормированного ($BM_{norm}$) изображений**

**Шаг 9**

Перейдем к извлечению скрытого сообщения.

Принятые обозначения:

- **BM*** — заполненный контейнер;
- **X*** и **Y*** — соответственно, размеры контейнера по вертикали (количество строк пикселей) и горизонтали (количество столбцов);
- **N*** — размерность выделяемого блока контейнера;
- $N^*_S$ — общее количество блоков;
- $T^*_1$, $T^*_2$ — значения первого и второго порогов сравнения;
- **μ*** — массив секретных масок.

Программные модули разбиения изображения на блоки (**SM***) и классификации пикселей блока на зоны (**Zone***) аналогичны, соответственно, модулям (М.34) и (М.35).

В отличие от операции встраивания, когда вычисление количества пикселей **n**, удовлетворяющих одной из возможных комбинаций зон и категорий, а также средних значений яркости λ каждой из четырех групп пикселей выполнялось непосредственно в программном модуле (см. (М.37)), при извлечении указанные массивы необходимо сформировать отдельно — (М.41), (М.42).

Сравнение средних значений яркости для каждой из зон (формула (5.13)) выполняем, используя модуль (М.43).

```
n* :=| for  s ∈ 1 .. N*_S
     |   Z ← Zone*_s
     |   м ← μ*_s                          (М.41)
     |   n ← (0 0)
     |        (0 0)
     |   (а) -- см. (М.37)
     |   n*_s ← n
     | n*
```

```
λ* :=| for  s ∈ 1 .. N*_S
     |   Б ← SM*_s
     |   Z ← Zone*_s
     |   м ← μ*_s                          (М.42)
     |   n ← n*_s
     |   Σλ ← (0 0)
     |          (0 0)
     |   (б) -- см. (М.37)
     |   λ*_s ← (Σλ / n)  (с стрелкой)
     | λ*
```

```
Σ :=| for  s ∈ 1 .. N*_S
    |   for  x ∈ 1 .. 2                    (М.43)
    |     E_x ← (λ*_s)_x,1 − (λ*_s)_x,2
    |   Σ_s ← E
    | Σ
```

Непосредственно модуль извлечения — (М.44), в котором реализована проверка условий, изложенных в теоретическом описании данного метода. Параметр **ε** выбирается достаточно близким к нулю (~ 0,05).

```
M* vec_bin :=| for  s ∈ 1 .. N*_S
             |   E ← Σ_s
             |   n ← n*_s
             |   if  E_1 · E_2 > ε
             |     M*_s ← 1  if  E_1 > 0
             |     M*_s ← 0  if  E_1 < 0
             |   if  E_1 · E_2 < ε                    (М.44)
             |     E' ← E_1 · (n_1,1 + n_1,2) + E_2 · (n_2,1 + n_2,2)
             |     M*_s ← 1  if  E' > 0
             |     M*_s ← 0  if  E' < 0
             |   if  −ε ≤ E_1 · E_2 ≤ ε
             |     E'' ← max(|E_1| , |E_2|)
             |     M*_s ← 1  if  E'' > 0
             |     M*_s ← 0  if  E'' < 0
             |     M*_s ← "0/1"  if  E'' = 0
             | M*
```

Вектор двоичных данных, возвращаемый модулем (М.44), преобразуется в строку символов аналогично тому, как это было при моделировании предыдущих методов.

Результаты, полученные при вычислении параметров визуального искажения, занесены в табл. 5.1 (стр. 125).

#### 5.3.2.9. Другие методы скрытия данных в пространственной области

Нетрадиционным является алгоритм, основанный на копировании блоков из одной случайно выбранной текстурной области в другую, которая имеет подобные статистические характеристики [14], что приводит к появлению в изображении полностью идентичных блоков. Эти блоки могут быть обнаружены таким образом:

- анализ автокорреляционной функции стеганоизображения и определение ее пиков;
- сдвиг изображения в соответствии с этими пиками и вычитание изображения из его сдвинутой копии;
- разница в местах размещения копированных блоков должна быть близкой к нулю. Поэтому можно выбрать некий порог и значения, не превышающие этот порог по абсолютной величине, считать искомыми блоками.

Поскольку копии блоков идентичны, они изменяются одинаково при преобразованиях всего изображения. Если сделать размер блоков достаточно большим, алгоритм будет стойким к большинству из негеометрических искажений. В проведенных авторами экспериментах была подтверждена стойкость алгоритма к фильтрации, компрессии, вращению изображения [14].

Основным недостатком алгоритма, по-видимому, является исключительная сложность нахождения достаточного количества областей, блоки из которых могут быть заменены без заметного ухудшения качества изображения. Кроме того, в данном алгоритме в качестве контейнера могут использоваться только достаточно текстурированные изображения.

Алгоритм, предложенный в [102], позволяет встраивать информацию в блоки 8×8 изображения-контейнера. В начале алгоритма создается маска $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{y})$, размерность которой отвечает размерности массива контейнера, а элементами являются псевдослучайно распределенные 0 и 1: $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{y}) \in \{0; 1\}$. Каждый блок $\boldsymbol{Б}$ в зависимости от значения элементов маски делится на два подмассива $\boldsymbol{Б}_1$ и $\boldsymbol{Б}_2$, для каждого из которых вычисляются средние значения яркости — $\boldsymbol{\lambda}_1$ и $\boldsymbol{\lambda}_2$. Бит скрываемого сообщения встраивается следующим образом:

$$\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1, \text{ при } \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2 > \boldsymbol{E}; \\ 0, \text{ при } \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2 < -\boldsymbol{E}, \end{cases} \tag{5.14}$$

где $\boldsymbol{E}$ — некоторое значение порога (необходимая разница между указанными средними значениями яркости).

В тех случаях, когда условие (5.14) не выполняется, соответствующим образом модифицируют значение яркости пикселей одного из подмассивов ($\boldsymbol{Б}_1$ или $\boldsymbol{Б}_2$). Для извлечения бита скрытого сообщения проводится вычисление соответствующих средних значений яркости подмассивов — $\boldsymbol{\lambda}^*_1$ і $\boldsymbol{\lambda}^*_2$. Разница между ними позволяет определить значения скрытого бита:

$$\boldsymbol{b_i} = \begin{cases} 1, \text{ при } \boldsymbol{\lambda}^*_1 - \boldsymbol{\lambda}^*_2 > 0; \\ 0, \text{ при } \boldsymbol{\lambda}^*_1 - \boldsymbol{\lambda}^*_2 < 0; \\ \boldsymbol{?} \text{ при } \boldsymbol{\lambda}^*_1 - \boldsymbol{\lambda}^*_2 = 0. \end{cases} \tag{5.15}$$

Таблица 5.1.

**Показатели визуального искажения в случае скрытия данных в пространственной области изображения**

| *Название показателя искажения* | *Оригинал* | *Методы скрытия в пространственной области* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | *НЗБ* | *ПС интервал* | *ПСП* | *Блочного кодирования* | *Замены палитры* | *Кванто-вания* | *Куттера-Джордана* | *Дармстедтера-Дэлэйгла* |
| Максимальная разность, ***MD*** | 0 | 1 | 1 | 1 | 1 | 3 | 3 | 38 | 54 |
| Средняя абсолютная разность, ***AD*** | 0 | 0.494 | $7.690 \cdot 10^{-3}$ | $5.920 \cdot 10^{-3}$ | $6.165 \cdot 10^{-3}$ | $9.827 \cdot 10^{-3}$ | $7.141 \cdot 10^{-3}$ | 4.588 | 17.704 |
| Нормированная средняя абсолютная разность, ***NAD*** | 0 | $3.823 \cdot 10^{-3}$ | $5.956 \cdot 10^{-5}$ | $4.585 \cdot 10^{-5}$ | $4.774 \cdot 10^{-5}$ | $7.611 \cdot 10^{-5}$ | $5.535 \cdot 10^{-5}$ | 0.050 | 0.137 |
| Среднеквадратическая ошибка, ***MSE*** | 0 | 0.494 | $7.690 \cdot 10^{-3}$ | $5.920 \cdot 10^{-3}$ | $6.165 \cdot 10^{-3}$ | 0.017 | $9.460 \cdot 10^{-3}$ | 235.708 | 456.887 |
| Нормированная среднеквадратическая ошибка, ***NMSE*** | 0 | $2.010 \cdot 10^{-5}$ | $3.132 \cdot 10^{-7}$ | $2.411 \cdot 10^{-7}$ | $2.510 \cdot 10^{-7}$ | $7.084 \cdot 10^{-7}$ | $3.853 \cdot 10^{-7}$ | $9.599 \cdot 10^{-3}$ | 0.019 |
| $\boldsymbol{L^p}$-норма, $\boldsymbol{p} = 2$ | 0 | 0.703 | 0.088 | 0.077 | 0.079 | 0.132 | 0.097 | 11.301 | 21.375 |
| Лапласова среднеквадратическая ошибка, ***LMSE*** | 0 | $9.815 \cdot 10^{-4}$ | $1.560 \cdot 10^{-5}$ | $1.263 \cdot 10^{-5}$ | $1.253 \cdot 10^{-5}$ | $1.990 \cdot 10^{-5}$ | $1.855 \cdot 10^{-5}$ | 0.240 | 0.420 |
| Отношение “сигнал/шум”, ***SNR*** | $\infty$ | $4.975 \cdot 10^{4}$ | $3.193 \cdot 10^{6}$ | $4.148 \cdot 10^{6}$ | $3.983 \cdot 10^{6}$ | $1.412 \cdot 10^{6}$ | $2.596 \cdot 10^{6}$ | 192.271 | 53.746 |
| Максимальное отношение “сигнал/шум”, ***PSNR*** | $\infty$ | $1.317 \cdot 10^{5}$ | $8.455 \cdot 10^{6}$ | $1.044 \cdot 10^{7}$ | $1.055 \cdot 10^{7}$ | $3.738 \cdot 10^{6}$ | $6.873 \cdot 10^{6}$ | 509.139 | 142.322 |
| Качество изображения, ***IF*** | 1 | 0.999980 | $\approx 1$ | $\approx 1$ | $\approx 1$ | 0.999999 | $\approx 1$ | 0.994799 | 0.981394 |
| Нормированная взаимная корреляция, ***NC*** | 1 | 0.999439 | 0.999992 | 0.999998 | 0.999988 | 0.999942 | 1.000001 | 0.988343 | 0.942705 |
| Качество корреляции, ***CQ*** | 190.182 | 190.076 | 190.181 | 190.182 | 190.180 | 190.172 | 190.183 | 187.966 | 179.286 |
| Структурное содержание, ***SC*** | 1 | 1.001103 | 1.000016 | 1.000004 | 1.000025 | 1.000114 | 0.999999 | 1.018447 | 1.106175 |
| Общее сигма-отношение “сигнал/шум”, ***GSSNR*** | $\infty$ | $1.298 \cdot 10^{5}$ | $9.751 \cdot 10^{6}$ | $8.026 \cdot 10^{6}$ | $5.306 \cdot 10^{6}$ | $5.797 \cdot 10^{5}$ | $3.648 \cdot 10^{6}$ | 187.522 | 31.555 |
| Сигма-отношение “сигнал/шум”, ***SSNR*** | $\infty$ | 142.5 | 62 | 42.4 | 39.7 | 7.6 | 19.5 | 41.8 | 57.3 |
| Нормированное отношение “сигма/ошибка”, ***NSER*** | 256 | 60 | 155 | 175 | 179 | 241 | 214 | 111 | 83.5 |
| Подобие гистограмм, ***HS*** | 0 | 3918 | 176 | 138 | 154 | 184 | 150 | 2068 | 10372 |

### *5.3.3. Скрытие данных в частотной области изображения*

Как уже было отмечено выше, стеганографические методы скрытия данных в пространственной области изображения являются нестойкими к большинству из известных видов искажений. Так, например, использование операции компрессии с потерями (относительно изображения, это может быть JPEG-компрессия) приводит к частичному или, что более вероятно, полному уничтожению встроенной в контейнер информации. Более стойкими к разнообразным искажениям, в том числе и компрессии, являются методы, использующие для скрытия данных не пространственную область контейнера, а частотную.

Существует несколько способов представления изображения в частотной области. При этом используется та или иная декомпозиция изображения, используемого в качестве контейнера. Например, существуют методы на основе использования дискретного косинусного преобразования (ДКП), дискретного преобразования Фурье (ДПФ), вейвлет-преобразования, преобразования Карунена-Лоева и некоторые другие. Подобные преобразования могут применяться либо к отдельным частям изображения, либо к изображению в целом.

Наибольшее распространение среди всех ортогональных преобразований в стеганографии получили вейвлет-преобразования и ДКП [5], что определенной мерой объясняется значительным распространением их использования при компрессии изображений. Кроме того, для скрытия данных целесообразно применять именно то преобразование изображения, которому последнее будет подвергаться со временем при возможной компрессии. Например, известно, что алгоритм ДКП является базовым в стандарте JPEG, а вейвлет-преобразования — в стандарте JPEG2000.

Стеганоалгоритм может быть достаточно стойким к последующей компрессии изображения, только если он будет учитывать особенности алгоритма перспективного сжатия. При этом, конечно, стеганоалгоритм, в основу которого заложено вейвлет-преобразование, совсем не обязательно будет стойким к дискретнокосинусному алгоритму сжатия, и наоборот. Авторы [5] отмечают, что еще большие трудности возникают при выборе метода стеганопреобразования во время скрытия данных в потоковом видео. Причина этого — одной из составляющих алгоритмов компрессии видеоинформации (в дополнение к компрессии неподвижного кадра), является кодирование векторов компенсации движения. При компрессии неподвижного изображения данная компенсация отсутствует за ненадобностью. Чтобы быть в достаточной степени стойким, стеганоалгоритм должен учитывать данный фактор.

Остается также открытым вопрос о существовании стойкого стеганопреобразования, которое было бы независимым от применяемого в дальнейшем алгоритма компрессии. Рассмотрение различных ортонормированных преобразований, таких как ДПФ, ДКП, преобразование Хартли, субполосное преобразование и др. с позиции теории информации проведено авторами [19, 44].

На сегодняшний день известно достаточно большое количество моделей, позволяющих оценить пропускную способность канала передачи скрытых данных. Ниже приведена одна из них, представленная в работах [5, 84].

Пусть $\boldsymbol{C}$ — первичное изображение (контейнер-оригинал), $\boldsymbol{M}$ — сообщение, которое подлежит скрытию. Тогда модифицированное изображение (стеганоконтейнер) $\boldsymbol{S} = \boldsymbol{C} + \boldsymbol{M}$. Также предполагается, что модифицированное изображение $\boldsymbol{S}$ визуально неотличимо от первичного и может быть подвергнуто в стеганоканале компрессии с потерями: $\boldsymbol{S}^{\nabla} = \boldsymbol{\Theta}(\boldsymbol{S})$, где $\boldsymbol{\Theta}(\bullet)$ — оператор компрессии.

Задача адресата — извлечь из полученного контейнера $S^{\nabla}$ встроенные на предыдущем этапе биты данных $M_i$.

Основное, что будет при этом интересовать нас, — ответ на вопрос: какое количество бит можно эффективно встраивать в изображение и со временем извлечь из него при условии удовлетворительно низкой вероятности ошибок на последнем этапе. Другими словами, какова пропускная способность канала передачи скрытых данных при условии наличия в канале связи определенного алгоритма компрессии?

Блок-схема такого стеганоканала представлена на рис. 5.23.

Сообщение $M$ передается по каналу, который имеет два источника «шума»: $C$ — изображение-контейнер и «шум» $\Theta$, возникающий в результате операций компрессии/декомпрессии. При этом $S^{\nabla}$ и $M^{*}$ — возможно искаженные стеганоконтейнер и, как результат, — оценка полезного сообщения.

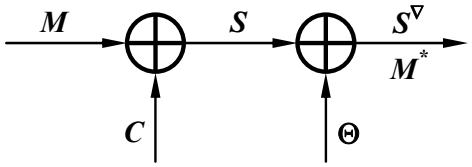



**Рис. 5.23. Блок-схема стеганоканала с атакой в виде компрессии**

Структурная схема стеганосистемы приведена на рис. 5.24.

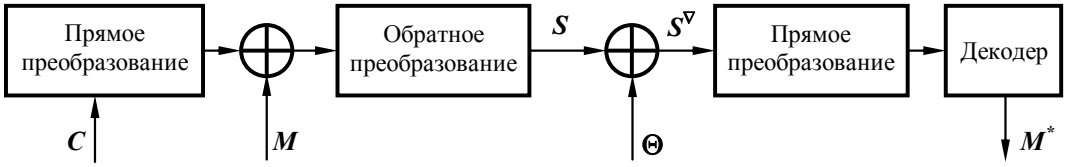



**Рис. 5.24. Структурная схема стеганосистемы при наличии в стеганоканале атаки компрессии**

Изображение $C$ раскладывается на $D$ субполос (прямое преобразование), в каждую из которых встраивается скрываемая информация $M$. После обратного преобразования получается модифицированное изображение $S$. После компрессии/ декомпрессии $\Theta$ в канале связи получается изображение $S^{\nabla}$, которое на принимающей стороне вновь подвергается прямому преобразованию и из каждой субполосы $D$ независимо извлекается скрытое сообщение — оценка $M^{*}$.

Как указывается в [5], реальные изображения не представляют собой случайные процессы с равномерно распределенными значениями величин. Известно, и данный факт используется в алгоритмах компрессии, что б*о*льшая часть энергии изображений сосредоточена в низкочастотной (НЧ) области спектра. Отсюда и возникает необходимость в осуществлении декомпозиции изображения на субполосы, к которым прибавляется стеганосообщение. НЧ субполосы содержат основную часть энергии изображения и, таким образом, носят шумовой характер. Высокочастотные (ВЧ) субполосы спектра изображения наибольшим образом поддаются влиянию со стороны разнообразных алгоритмов обработки, таких как, например, компрессия или НЧ-фильтрация. Таким образом, можно сделать вывод, что для встраивания сообщения самыми оптимальными являются среднечастотные (СЧ) субполосы спектра изображения. Типичное распределение шума изображения и шума обработки по спектру частоты изображено на рис. 5.25 [44, 85].

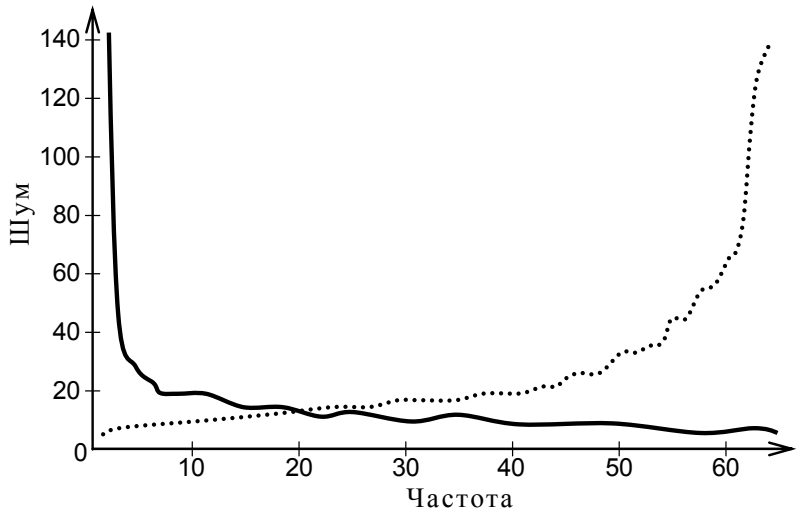



**Рис. 5.25. Зависимость шума изображения (сплошная линия) и шума обработки (пунктирная линия) от частоты**

Стеганографический канал можно разложить на ряд независимых подканалов. Такое разложение совершается за счет выполнения прямого и обратного преобразований. В каждом из $D$ подканалов существует по два источника шума. Пусть $\sigma^2_{C,\Theta_j}$, при $j = 1, ..., D$ — дисперсия коэффициентов преобразования (шум изображения) в каждом из подканалов. Тогда выражение для пропускной способности канала стеганосистемы приобретет вид:

$$B = \frac{X \cdot Y}{2 \cdot D} \cdot \sum_{j=1}^{D} \log_2 \left( 1 + \frac{\nu_j^2}{\sigma^2_{C_j} + \sigma^2_{\Theta_j}} \right) \text{ бит}, \qquad (5.16.)$$

где $\nu_j$ — визуальный порог для $j$-й субполосы ( $\nu_j^2$ — максимально допустимая энергия стеганосообщения, исходя из требований сохранения визуального качества изображения); $X$ и $Y$ — пиксельный размер изображения-контейнера.

Выбор значения визуального порога базируется на учете свойств ЗСЧ. Известно, что шум в ВЧ областях изображения более приемлемый, чем в НЧ областях. Можно вести некоторые весовые коэффициенты: $\nu_j^2 = \kappa \cdot \sigma^{2 \cdot \alpha}_{C,\Theta_j}$, где $0 \le \alpha \le 1$, а $\kappa << \sigma_{C,\Theta_j} \ \forall \ j$ — константа.

Случай, когда $\alpha = 0$, отвечает равномерному распределению стеганограммы по всем субполосам. Случай $\alpha = 1$ отвечает распределению стеганограммы в соответствии с дисперсией субполос.

После некоторых упрощений можно получить выражение для пропускной способности канала передачи скрытых данных:

$$B = \frac{X \cdot Y}{2 \cdot D} \cdot \sum_{j=1}^{D} \log_2 \left( 1 + \frac{\kappa_1 \cdot \sigma^{2 \cdot \alpha}_{C_j}}{\sigma^2_{C_j}} \right) \approx \frac{X \cdot Y}{2 \cdot D} \cdot \log_2 \left( 1 + \sum_{j=1}^{D} \frac{\kappa_1}{\sigma^{2 \cdot (1-\alpha)}_{C_j}} \right). \qquad (5.17)$$

Знак приближения в выражении (5.17) является справедливым, поскольку $\kappa_1 \cdot \sigma_{C_j}^{2\cdot\alpha} \Big/ \sigma_{C_j}^2 << 1$ для любого значения $j$. Очевидно, что при $\alpha = 1$ декомпозиция никоим образом не будет влиять на пропускную способность стеганоканала. При $\alpha < 1$ пропускная способность будет возрастать за счет того, что в области с низкой дисперсией (высокочастотной области) стеганосигналу прибавляется относительно больше энергии.

В [85] приведены результаты многочисленных экспериментов, которые позволили авторам дать определенные рекомендации относительно выбора вида преобразования для стеганографии. Известно, что преобразования можно упорядочить по достижимым выигрышам от алгоритма кодирования (рис. 5.26).

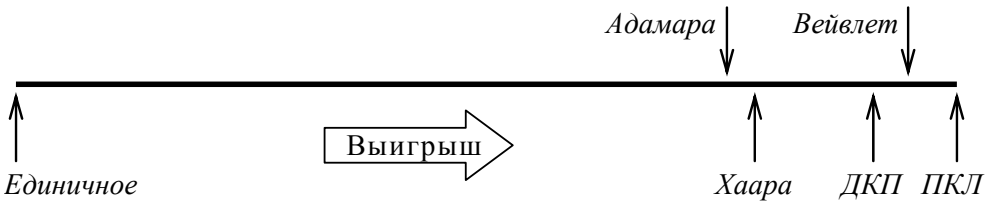



**Рис. 5.26. Виды преобразований, упорядоченные по достижимым выигрышам от использования алгоритма кодирования**

Под выигрышем от кодирования подразумевается степень перераспределения дисперсий коэффициентов преобразования. Наибольший выигрыш дает преобразование Карунена-Лоева (ПКЛ), наименьший — разложение за базисом единичного импульса (то есть отсутствие преобразования).

Преобразования, которые характеризуются высокими значениями выигрыша от кодирования, такие как ДКП, вейвлет-преобразования, характеризуются резко неравномерным распределением дисперсий коэффициентов субполос. Высокочастотные субполосы не подходят для встраивания из-за большого шума обработки, а низкочастотные — из-за высокого шума изображения (см. рис. 5.25). Поэтому, как отмечалось в [85], приходится ограничиваться среднечастотными полосами, у которых шум изображения приблизительно равен шуму обработки. Поскольку таких полос немного, то пропускная способность стеганоканала является сравнительно малой.

В случае применения преобразования с более низким выигрышем от кодирования, например, преобразования Адамара или Фурье, существует больше блоков, у которых шум изображения приблизительно равняется шуму обработки, а, следовательно, и пропускная способность выше. Вывод, к которому пришли авторы указанной работы, достаточно неожиданный: для повышения пропускной способности стеганографического канала целесообразно применять преобразования с меньшими выигрышами от кодирования, которые плохо подходят для компрессии сигналов.

Эффективность применения вейвлет-преобразования и ДКП для компрессии изображений объясняется тем, что они хорошо моделируют процесс обработки изображения в ЗСЧ, отделяя существенные детали от второстепенных. Таким образом, данные преобразования более целесообразно использовать в случае присутствия активного нарушителя, поскольку модификация значимых коэффициентов может привести к неприемлемому искажению изображения.

При применении преобразований с низкими значениями выигрыша от кодирования существует значительная опасность нарушения встроенных данных, по причине того, что коэффициенты преобразования менее стойки к модификациям. Одна-

ко при этом существует бо́льшая гибкость в выборе преобразования, и если последнее неизвестно нарушителю (хотя это и противоречит принципу Керхгофса), то модифицировать стеганограмму будет существенно сложнее.

Во время цифровой обработки изображения часто применяется двумерная версия дискретного косинусного преобразования:

$$\Omega(\upsilon,\nu)=\frac{\zeta(\upsilon)\cdot\zeta(\nu)}{\sqrt{2N}}\cdot\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}C(x,y)\cdot\cos\left[\frac{\pi\cdot\upsilon\cdot(2x+1)}{2N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2y+1)}{2N}\right];\qquad(5.18\ a)$$

$$S(x,y)=\frac{1}{\sqrt{2N}}\cdot\sum_{\upsilon=0}^{N-1}\sum_{\nu=0}^{N-1}\zeta(\upsilon)\cdot\zeta(\nu)\cdot\Omega(\upsilon,\nu)\cdot\cos\left[\frac{\pi\cdot\upsilon\cdot(2x+1)}{2N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2y+1)}{2N}\right],\quad(5.18,\ б)$$

где $C(x, y)$ и $S(x, y)$ — соответственно, элементы оригинального и восстановленного по коэффициентам ДКП изображения размерностью $N\times N$; $x, y$ — пространственные координаты пикселей изображения; $\Omega(\upsilon, \nu)$ — массив коэффициентов ДКП; $\upsilon, \nu$ — координаты в частотной области; $\zeta(\upsilon) = 1/\sqrt{2}$, если $\upsilon = 0$, и $\zeta(\upsilon) = 1$, если $\upsilon > 0$.

Рассмотрим существующие методы, которые базируются на алгоритме ДКП.

#### 5.3.3.1. Метод относительной замены величин коэффициентов ДКП (метод Коха и Жао)

Один из наиболее распространенных на сегодня методов скрытия конфиденциальной информации в частотной области изображения заключается в ***относительной замене величин коэффициентов ДКП***, который в свое время описали Кох (E. Koch) и Жао (J. Zhao) [86, 87].

На начальном этапе первичное изображение разбивается на блоки размерностью 8×8 пикселей. ДКП применяется к каждому блоку — формула (5.18 *а*), в результате чего получают матрицы 8×8 коэффициентов ДКП, которые зачастую обозначают $\Omega_b(\upsilon,\nu)$, где $b$ — номер блока контейнера $C$, а $(\upsilon,\nu)$ — позиция коэффициента в этом блоке. Каждый блок при этом предназначен для скрытия одного бита данных.

Было предложено две реализации алгоритма: псевдослучайно могут выбираться два или три коэффициента ДКП. Рассмотрим первый вариант.

Во время организации секретного канала абоненты должны предварительно договориться о двух конкретных коэффициентах ДКП из каждого блока, которые будут использоваться для скрытия данных. Зададим данные коэффициенты их координатами в массивах коэффициентов ДКП: $(\upsilon_1, \nu_1)$ и $(\upsilon_2, \nu_2)$. Кроме этого, указанные коэффициенты должны отвечать косинус-функциям со средними частотами, что обеспечит скрытость информации в существенных для ЗСЧ областях сигнала, к тому же информация не будет искажаться при JPEG-компрессии с малым коэффициентом сжатия.

Непосредственно процесс скрытия начинается со случайного выбора блока $C_b$ изображения, предназначенного для кодирования $b$-го бита сообщения. Встраивание информации осуществляется таким образом: для передачи бита «0» стремятся, чтобы разница абсолютных значений коэффициентов ДКП превышала некоторую положительную величину, а для передачи бита «1» эта разница делается меньшей по сравнению с некоторой отрицательной величиной:

$$\begin{cases}\left|\Omega_b(\upsilon_1,\nu_1)\right|-\left|\Omega_b(\upsilon_2,\nu_2)\right|>P,\ \text{при}\ m_b=0;\\ \left|\Omega_b(\upsilon_1,\nu_1)\right|-\left|\Omega_b(\upsilon_2,\nu_2)\right|<-P,\ \text{при}\ m_b=1.\end{cases}\qquad(5.19)$$

Таким образом, первичное изображение искажается за счет внесения изменений в коэффициенты ДКП, если их относительная величина не отвечает скрываемому биту. Чем больше значение ***P***, тем стеганосистема, созданная на основе данного метода, является более стойкой к компрессии, однако качество изображения при этом значительно ухудшается.

После соответствующего внесения коррекции в значения коэффициентов, которые должны удовлетворять неравенству (5.19), проводится обратное ДКП.

Для извлечения данных, в декодере выполняется аналогичная процедура выбора коэффициентов, а решение о переданном бите принимается в соответствии со следующим правилом:

$$\begin{cases} m_b^* = 0, \text{ при } \left|\Omega_b^*(\upsilon_1,\nu_1)\right| > \left|\Omega_b^*(\upsilon_2,\nu_2)\right|; \\ m_b^* = 1, \text{ при } \left|\Omega_b^*(\upsilon_1,\nu_1)\right| < \left|\Omega_b^*(\upsilon_2,\nu_2)\right|. \end{cases} \quad (5.20)$$

Промоделируем описанный метод в программе MathCAD.

**Шаг 1**

Выделяем массивы цветовых компонентов изображения:

**R** := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

В связи с низкой чувствительностью ЗСЧ к каналу синего цвета и возможным при определенных обстоятельствах довольно значительным искажением контейнера при встраивании, секретное сообщение будем встраивать в массив **B**.

Определим размерность массива **B** и зададим размерность сегментов (блоков), на которые он будет разбиваться: количество строк **X** := **rows**(**B**), **X** = 128; количество столбцов **Y** := **cols**(**B**), **Y** = 128; размерность сегментов **N** := 8 пикселей.

Общее количество сегментов, на которое разбивается контейнер: $\mathbf{N_C} := \mathbf{X} \cdot \mathbf{Y} / \mathbf{N}^2$, $\mathbf{N_C}$ = 256 сегментов.

**Шаг 2**

Разбивку массива **B** на сегменты **C** проводим с помощью модуля (М.45).

```
C := | c1 ← 1
     | c2 ← N
     | for b ∈ 1 .. N_C
     |     | r1 ← mod[N·(b − 1) + 1 , X]
     |     | r2 ← r1 + N − 1
     |     | C_b ← submatrix(B , r1 , r2 , c1 , c2)
     |     | if r2 = X
     |     |     | c1 ← c1 + N
     |     |     | c2 ← c2 + N
     | C
```

(М.45)

Каждый сегмент **C** предназначен для скрытия одного бита конфиденциального сообщения **M**. Поэтому предварительно необходимо проверить достаточность количества сегментов для этой операции.

Пусть, как и в предыдущих случаях, сообщение имеет вид **M** := "© Пузыренко А.Ю., 2005 г.". При этом количество бит в скрываемом сообщении (один символ со-

общения кодируется одним байтом): $L_M := 8 \cdot \mathbf{strlen}(M) = 200$ бит $< N_C = 256$. Итак, объем сообщения является приемлемым для встраивания.

**Шаг 3**

Применим к каждому из сегментов прямое дискретное косинусное преобразование — см. выражение (5.18 *а*). Программный модуль прямого ДКП состоит из двух частей: первая (М.46) определяет значения коэффициентов $\zeta$ для текущего значения аргумента $\chi$, вторая (М.47) проводит вычисление спектральных коэффициентов ДКП для каждого сегмента $C_b$, при этом возвращается соответствующая матрица размерностью $N \times N$.

Коэффициент в левом верхнем углу матрицы $\Omega_b$ (напомним, что в нашем случае нумерация элементов массивов начинается с единицы) — $(\Omega_b)_{1,1}$ содержит информацию о яркости всего сегмента (его зачастую называют DC-коэффициентом). Другие коэффициенты называются AC-коэффициентами. Отметим также, что коэффициенты НЧ компонентов размещены ближе к левому верхнему углу, а ВЧ компонентов — ближе к правому нижнему углу (рис. 5.27).

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | –603 | 203 | 11 | 45 | –30 | –14 | –14 | –7 |
| 2 | –108 | –93 | 10 | 49 | 27 | 6 | 8 | 2 |
| 3 | –42 | –20 | –6 | 16 | 17 | 9 | 3 | 3 |
| 4 | 56 | 69 | 7 | –25 | –10 | –5 | –2 | –2 |
| 5 | –33 | –21 | 17 | 8 | 3 | –4 | –5 | –3 |
| 6 | –16 | –14 | 8 | 2 | –4 | –2 | 1 | 1 |
| 7 | 0 | –5 | –6 | –1 | 2 | 3 | 1 | 1 |
| 8 | 9 | 5 | –6 | –9 | 0 | 3 | 3 | 2 |

– НЧ компоненты;

– СЧ компоненты;

– ВЧ компоненты.

**Рис. 5.27. Пример массива Ω коэффициентов ДКП**

$$\zeta(\chi) := \begin{cases} \frac{1}{\sqrt{2}} & \text{if } \chi = 0 \\ 1 & \text{if } \chi > 0 \end{cases} \qquad \text{(М.46)}$$

$$\Omega := \left| \begin{array}{l} \text{for } b \in 1..N_C \\ \quad \left| \begin{array}{l} \text{for } u \in 0..N-1 \\ \quad \text{for } v \in 0..N-1 \\ \quad\quad \Omega'_{u+1,v+1} \leftarrow \frac{\zeta(u)\cdot\zeta(v)}{\sqrt{2\cdot N}} \times \\ \quad\quad \times \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left[ (C_b)_{x+1,y+1} \cdot \cos\left[\frac{\pi\cdot u\cdot(2\cdot x+1)}{2\cdot N}\right] \cdot \cos\left[\frac{\pi\cdot v\cdot(2\cdot y+1)}{2\cdot N}\right] \right] \\ \quad \Omega_b \leftarrow \Omega' \end{array} \right. \\ \Omega \end{array} \right. \qquad \text{(М.47)}$$

**Шаг 4**

Как уже отмечалось выше, НЧ компоненты содержат преобладающую часть энергии изображения и, следовательно, носят шумовой характер. ВЧ компоненты больше поддаются влиянию со стороны различных алгоритмов обработки (см.

рис. 5.25). Таким образом, для встраивания сообщения оптимальными являются среднечастотные компоненты спектра изображения.

Зададим позиции двух коэффициентов ДКП в массиве $\Omega_b$, которые будут использоваться при встраивании и извлечении сообщения в/из контейнера. Для создания более защищенной от взлома стеганосистемы отмеченная пара коэффициентов ДКП должна выбираться псевдослучайно, конечно, из множества среднечастотных коэффициентов. Пусть первый коэффициент определяется координатами ($\upsilon_1 := 4$; $\nu_1 := 5$), а второй — ($\upsilon_2 := 5$; $\nu_2 := 4$).

Также установим значение порога, с которым будут сравниваться результаты разности модулей коэффициентов. Пусть $P := 25$.

**Шаг 5**

Встраивание информации проведем в соответствии с рекомендациями, которые даны перед выражением (5.19): для передачи нулевого бита необходимо, чтобы разность модулей коэффициентов ДКП превышала величину **P**, а для передачи единичного бита эта разность должна быть меньше **–P**. Данный принцип положен в основу модуля (М.48).

```
Ω_M := | Ω_M ← Ω
       | M ← str2vec(M)
       | b ← 1
       | for μ ∈ 1 .. rows(M)
       |     m ← D2B(M_μ)
       |     for i ∈ 1 .. 8
       |         Ω' ← Ω_{i+N·(μ−1)}
       |         ϖ1 ← |Ω'_{υ1, ν1}|
       |         ϖ2 ← |Ω'_{υ2, ν2}|
       |         z1 ← 1  if Ω'_{υ1, ν1} ≥ 0
       |         z1 ← −1  if Ω'_{υ1, ν1} < 0
       |         z2 ← 1  if Ω'_{υ2, ν2} ≥ 0
       |         z2 ← −1  if Ω'_{υ2, ν2} < 0
       |         ϖ1 ← P + ϖ2 + 1  if ϖ1 − ϖ2 ≤ P  if m_i = 0
       |         ϖ2 ← P + ϖ1 + 1  if ϖ1 − ϖ2 ≥ −P  if m_i = 1
       |         Ω'_{υ1, ν1} ← z1 · ϖ1
       |         Ω'_{υ2, ν2} ← z2 · ϖ2
       |         Ω_{M_b} ← Ω'
       |         b ← b + 1
       | Ω_M
```

(М.48)

В начале модуля массиву $\Omega_M$ присваивается значение массива $\Omega$. Строка символов **M** преобразуется в вектор их ASCII-кодов. В цикле изменения $\mu$ (перебор кодов символов) формат кода каждого символа преобразуется из десятичного в двоичный. Каждый из восьми полученных при этом бит скрывается в отдельном сегменте изо-

бражения путем модификации значений коэффициентов ДКП соответствующего сегмента. Для уменьшения заметности встраивания, вместо одностороннего изменения одного из двух коэффициентов ДКП для удовлетворения требований (5.19), можно изменять их одновременно в противоположных направлениях: например, при встраивании «0» — увеличивать модуль коэффициента с координатами $(\upsilon_1, \nu_1)$ и одновременно уменьшать модуль коэффициента координатами $(\upsilon_2, \nu_2)$.

**Шаг 6**

Выполним обратное ДКП (М.49) в соответствии с формулой (5.18 *б*).

$$
\begin{aligned}
&C_M := \Big| \text{for } b \in 1..N_C \\
&\quad \Big| \text{for } x \in 0..N-1 \\
&\quad\quad \text{for } y \in 0..N-1 \\
&\quad\quad\quad C'M_{x+1,y+1} \leftarrow \frac{1}{\sqrt{2\cdot N}} \cdot \sum_{\upsilon=0}^{N-1} \sum_{\nu=0}^{N-1} \left[ \zeta(\upsilon)\cdot\zeta(\nu)\cdot\left(\Omega M_b\right)_{\upsilon+1,\nu+1} \times \right. \\
&\quad\quad\quad\quad \left. \times \cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \\
&\quad C_{M_b} \leftarrow C'M \\
&C_M
\end{aligned}
\qquad \text{(M.49)}
$$

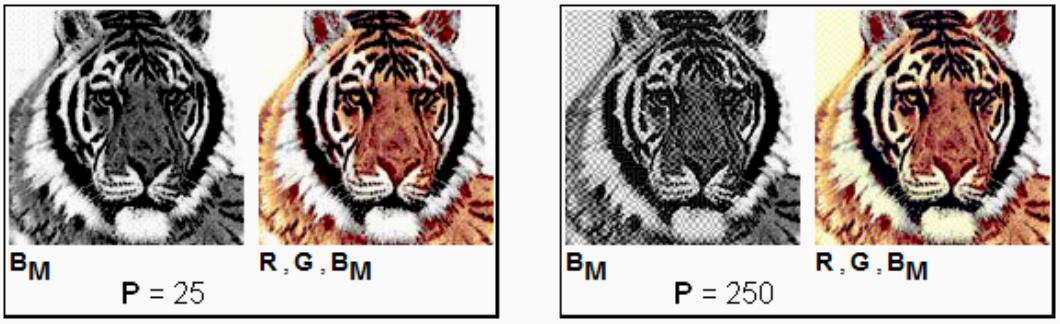
**Шаг 7**

Проведем сбор сегментов в массив, используя программный модуль (М.50).

$$
\begin{aligned}
&B_M := \Big| B_M \leftarrow C_{M_1} \\
&\text{for } b \in 2..X \div N \\
&\quad B_M \leftarrow \text{stack}\left(B_M, C_{M_b}\right) \\
&B'_M \leftarrow 0 \\
&\text{for } b \in X \div N + 1..N_C \\
&\quad \Big| B'_M \leftarrow C_{M_b} \quad \text{if } B'_M = 0 \\
&\quad B'_M \leftarrow \text{stack}\left(B'_M, C_{M_b}\right) \text{ otherwise} \\
&\quad \text{if } \text{mod}(b, X \div N) = 0 \\
&\quad\quad \Big| B_M \leftarrow \text{augment}\left(B_M, B'_M\right) \\
&\quad\quad B'_M \leftarrow 0 \\
&B_M \leftarrow \frac{B_M + \left|\min\left(B_M\right)\right|}{\max\left(B_M + \left|\min\left(B_M\right)\right|\right)} \cdot 255
\end{aligned}
\qquad \text{(M.50)}
$$

Поскольку модификация коэффициентов ДКП в некоторых случаях приводит к выходу значений интенсивностей пикселей изображения за пределы допустимого диапазона [0, 255], в конце модуля (М.50) проводим нормирование указанных значений.

Воссоздаем массив восстановленного изображения графически (рис. 5.28 *а*). Также на рис. 5.28 изображен контейнер-результат, восстановленный по цветовым составляющим $\mathbf{R\text{-}G\text{-}B_M}$. Кроме того, представлено вид, который приобретают массив синего цвета и контейнер-результат в случае установления значения порога $\mathbf{P} := 250$ при неизменных значениях ($\upsilon_1$; $\nu_1$) и ($\upsilon_2$; $\nu_2$) (рис. 5.28 *б*).




*а)* *б)*

**Рис. 5.28. Результаты встраивания в контейнер секретного сообщения M путем модификации коэффициентов ДКП сегментов массива канала синего при различных значениях порога P**

**Шаг 8**

Рассмотрим процесс извлечения сообщения на принимающей стороне.

Получателю должны быть известны: алгоритм скрытия, цветовой массив-контейнер (**B***) и его размерность (**X*** и **Y***), размерность сегментов (**N***) и матричные координаты коэффициентов косинусных функций, которые использовались для скрытия [($\upsilon^*_1$; $\nu^*_1$) и ($\upsilon^*_2$; $\nu^*_2$)].

Выполним вычисление общего количества сегментов в изображении-контейнере: $\mathbf{N^*_C} := \mathbf{X^*}\cdot\mathbf{Y^*}/\mathbf{N^{*2}}$.

Разбивка массива **B*** на сегменты выполняется с помощью модуля, аналогичного программному модулю (М.45). В результате получаем массив **C***, каждый сегмент которого представляет собой матрицу размерностью $\mathbf{N^*}\times\mathbf{N^*}$.

К каждому сегменту применяем прямое ДКП (см. модули (М.46), (М.47)), при этом получаем массив **Ω*** коэффициентов ДКП каждого отдельного сегмента **C***.

С помощью модуля (М.51) осуществляем извлечение скрытой информации. В основу модуля положена проверка соотношений (5.20), по результатам которой извлекаемому биту сообщения присваивается значение 0 или 1.

После извлечения каждого восьмого бита полученный вектор **m*** преобразуется из массива двоичных данных в соответствующее десятичное число, которое, в свою очередь, присваивается определенному переменной **j** элементу вектора **M***. Результат извлечения (вектор десятичных чисел) в конце модуля трансформируется из вектора ASCII-кодов в строку символов.

Результаты вычисления показателей визуального искажения графического контейнера при значениях порога встраивания $\mathbf{P} = 0.5$ и $\mathbf{P} = 25$ сведены в табл.5.4 (стр. 180).

#### 5.3.3.2. Метод Бенгама-Мемона-Эо-Юнг

Бенгам (D. Benham), Мемон (N. Memon), Эо (B.-L. Yeo) и Юнг (Minerva Yeung) [103] предложили оптимизированную версию вышерассмотренного метода. Причем оптимизация была проведена ими по двум направлениям: во-первых, было предло-

жено для встраивания использовать не все блоки, а только наиболее подходящие для этого, во-вторых, в частотной области блока для встраивания выбираются не два, а три коэффициента ДКП, что, как будет показано в дальнейшем, существенно уменьшает визуальные искажения контейнера. Рассмотрим отмеченные усовершенствования более подробно.

```
M* := | i ← 1
      | j ← 1
      | for μ ∈ 1 .. N*C
      |     Ω' ← Ω*μ
      |     ϖ1 ← |Ω'υ*1, ν*1|
      |     ϖ2 ← |Ω'υ*2, ν*2|
      |     m*i ← 0  if  ϖ1 > ϖ2
      |     m*i ← 1  if  ϖ1 < ϖ2
      |     i ← i + 1  if  rows(m*) < 8
      |     if  rows(m*) = 8
      |         | i ← 1
      |         | M*j ← B2D(m*)
      |         | m* ← 0
      |         | j ← j + 1
      | vec2str(M*)
```

(M.51)

Пригодными для встраивания информации считаются такие блоки изображения, которые одновременно удовлетворяют следующим двум требованиям:

- блоки не должны иметь резких переходов яркости;
- блоки не должны быть слишком монотонными.

Блоки, которые не отвечают первому требованию, характеризуются наличием слишком больших значений низкочастотных коэффициентов ДКП, сопоставимых по своей величине с DC-коэффициентом. Для блоков, которые не удовлетворяют второму требованию, характерно равенство нулю большинства высокочастотных коэффициентов. Указанные особенности являются критерием отбраковки непригодных блоков.

Отмеченные требования отбраковки учитываются использованием двух пороговых коэффициентов: $\boldsymbol{P}_{\mathbf{L}}$ (для первого требования) и $\boldsymbol{P}_{\mathbf{H}}$ (для второго требования), превышение ($\boldsymbol{P}_{\mathbf{L}}$) или недостижение ($\boldsymbol{P}_{\mathbf{H}}$) которых будет указывать на то, что рассматриваемый блок не пригоден для модификации в частотной области.

Встраивание в блок бита сообщения совершается следующим образом. Выбираются (для большей стойкости стеганосистемы — псевдослучайно) три коэффициента ДКП блока из среднечастотной области с координатами ($\upsilon_1$, $\nu_1$), ($\upsilon_2$, $\nu_2$) и ($\upsilon_3$, $\nu_3$). Если необходимо провести встраивание «0», эти коэффициенты изменяются таким образом (если, конечно, это необходимо), чтобы третий коэффициент стал

меньше любого из первых двух; если необходимо скрыть «1», он делается большим по сравнению с первым и вторым коэффициентами:

$$\begin{cases} \left.\begin{array}{l} |\Omega_b(\upsilon_3,\nu_3)|<|\Omega_b(\upsilon_1,\nu_1)|; \\ |\Omega_b(\upsilon_3,\nu_3)|<|\Omega_b(\upsilon_2,\nu_2)|. \end{array}\right\}, \text{ при } m_b=0; \\ \left.\begin{array}{l} |\Omega_b(\upsilon_3,\nu_3)|>|\Omega_b(\upsilon_1,\nu_1)|; \\ |\Omega_b(\upsilon_3,\nu_3)|>|\Omega_b(\upsilon_2,\nu_2)|. \end{array}\right\}, \text{ при } m_b=1. \end{cases} \quad (5.21)$$

Как и в предыдущем методе, для принятия решения о достаточности различения указанных коэффициентов ДКП, в выражение (5.21) вводится значение порога различения $P$:

$$\begin{cases} |\Omega_b(\upsilon_3,\nu_3)| < \min\left(|\Omega_b(\upsilon_1,\nu_1)|,|\Omega_b(\upsilon_2,\nu_2)|\right) - P, \text{ при } m_b=0; \\ |\Omega_b(\upsilon_3,\nu_3)| > \max\left(|\Omega_b(\upsilon_1,\nu_1)|,|\Omega_b(\upsilon_2,\nu_2)|\right) + P, \text{ при } m_b=1. \end{cases} \quad (5.22)$$

В том случае, если такая модификация приводит к слишком большой деградации изображения, коэффициенты не изменяют, и блок в качестве контейнера не используется.

Использование трех коэффициентов вместо двух и, что самое главное, отказ от модификации блоков изображения в случае неприемлемых их искажений, уменьшает погрешности, которые вносятся сообщением. Получатель всегда может определить блоки, в которые не проводилось встраивание, просто повторив анализ, аналогичный выполненному на передающей стороне.

Приведем возможный вариант реализации данного метода в программе MathCAD.

**Шаг 1**

Исходные данные, программные модули разбития массива контейнера на блоки и проведение операции прямого ДКП аналогичны использованным при моделировании предыдущего метода ((М.45)–(М.47)).

**Шаг 2**

Зададим координаты трех коэффициентов ДКП в массиве $\Omega_b$, которые будем использовать для встраивания и извлечения битов сообщения в/из контейнера. Например, ($\upsilon_1 := 7$; $\nu_1 := 3$), ($\upsilon_2 := 5$; $\nu_2 := 5$), ($\upsilon_3 := 3$; $\nu_3 := 7$).

Устанавливаем значения порогов отбраковки блоков. Пусть $P_L := 2600$, а $P_H := 40$. Также задаем порог различения $P := 50$.

**Шаг 3**

Встраивание сообщения $M$ в блоки контейнера будем выполнять, используя программный модуль (М.52).

В начале вычисления массиву $\Omega_M$ присваивается значение немодифицированного массива коэффициентов ДКП для всех блоков изображения — $\Omega$.

После преобразования формата сообщения из строки символов в вектор двоичных данных размерностью $L_M \times 1$ происходит непосредственно модификация оптимальных для встраивания блоков.

$\Omega_M :=$ | $\Omega_M \leftarrow \Omega$

Ⓘ -- см. (M.21), кроме $S \leftarrow C$

$w \leftarrow 0$

**for** $j \in 1 .. L_M$

$w \leftarrow w + 1$

**if** $w > N_C$

$\Omega_M \leftarrow$ **concat**("Встроено", **num2str**$(j-1)$, "бит.", "Расширьте границы")

**break**

**for** $c \in w .. N_C$

$\Omega' \leftarrow \Omega_c$

$\Sigma_{LF} \leftarrow 0$

$v\$ \leftarrow N - 1$

**for** $\upsilon \in 1 .. N - 1$

**for** $v \in 1 .. v\$$

$\Sigma_{LF} \leftarrow \Sigma_{LF} + |\Omega'_{\upsilon, v}|$ **if** $(\upsilon + v) > 2$

$v\$ \leftarrow v\$ - 1$ **if** $v = v\$$

$\Sigma_{HF} \leftarrow 0$

$v\$ \leftarrow N$

**for** $\upsilon \in 3 .. N$

**for** $v \in v\$ .. N$

$\Sigma_{HF} \leftarrow \Sigma_{HF} + |\Omega'_{\upsilon, v}|$

$v\$ \leftarrow v\$ - 1$ **if** $v = v\$$

**if** $\Sigma_{LF} < P_L \wedge \Sigma_{HF} > P_H$

$w \leftarrow c$

**break**

$\varpi_1 \leftarrow \Omega'_{\upsilon_1, v_1}$

$\varpi_2 \leftarrow \Omega'_{\upsilon_2, v_2}$

$\varpi_3 \leftarrow \Omega'_{\upsilon_3, v_3}$

**if** $\varpi_3 > \varpi_1 \vee \varpi_3 > \varpi_2$ **if** $M_{vec\_bin_j} = 0$

$\varpi_{min} \leftarrow \min\left(\begin{pmatrix} \varpi_1 \\ \varpi_2 \end{pmatrix}\right)$

$\varpi_3 \leftarrow \varpi_{min} - \frac{P}{2}$

$\varpi_1 \leftarrow \varpi_1 + \frac{P}{2}$ **if** $\varpi_1 = \varpi_{min}$

$\varpi_2 \leftarrow \varpi_2 + \frac{P}{2}$ **if** $\varpi_2 = \varpi_{min}$

$\varpi_3 \leftarrow \varpi_{min} - P$

**if** $\varpi_3 < \varpi_1 \vee \varpi_3 < \varpi_2$ **if** $M_{vec\_bin_j} = 1$

$\varpi_{max} \leftarrow \max\left(\begin{pmatrix} \varpi_1 \\ \varpi_2 \end{pmatrix}\right)$

$\varpi_3 \leftarrow \varpi_{max} + \frac{P}{2}$

$\varpi_1 \leftarrow \varpi_1 - \frac{P}{2}$ **if** $\varpi_1 = \varpi_{max}$

$\varpi_2 \leftarrow \varpi_2 - \frac{P}{2}$ **if** $\varpi_2 = \varpi_{max}$

$\varpi_3 \leftarrow \varpi_{max} + P$

$\Omega'_{\upsilon_1, v_1} \leftarrow \varpi_1$

$\Omega'_{\upsilon_2, v_2} \leftarrow \varpi_2$

$\Omega'_{\upsilon_3, v_3} \leftarrow \varpi_3$

$\Omega M_w \leftarrow \Omega'$

$\Omega_M$

(M.52)

Переменная-счетчик **w** перед началом цикла перебора элементов вектора $M_{vec\_bin}$ принимает нулевое значение. В начале цикла совершается проверка условия невыхода переменной **w** за границы общего количества сегментов $N_C$. В случае, если $w > N_C$, цикл прерывается после присвоения переменной $\Omega_M$ строки символов, которая информирует отправителя о количестве встроенных бит (при текущих значениях порогов $P_L$ и $P_H$) и рекомендует расширить границы (то есть увеличить значения $P_L$ и/или уменьшить $P_H$), что приведет к увеличению количества блоков, которые будут отвечать поставленным требованиям.

Разумеется, расширение границ приведет к появлению в списке блоков, выбранных для встраивания, не вполне оптимальных для данной операции. Поэтому в процессе договоренности между сторонами скрытого обмена относительно алгоритма псевдослучайного выбора трех пар координат коэффициентов ДКП и значений порогов отбраковки обязательно должна присутствовать фаза проверки выбранного в качестве контейнера изображения (набора изображений) на достаточность его пропускной способности. Другими словами, после установления определенных значений порогов $P_L$ и $P_H$, вычисляется количество блоков, признанных пригодными для встраивания, и выполняется оценка визуального искажения контейнера. По полученным результатам принимается решение о достаточности выбранных значений порогов или же о необходимости их изменения.

Например, установленные выше параметры позволяют признать пригодными для встраивания 201 блок в выбранном нами контейнере. Приведем результаты вычисления количества пригодных блоков $N_{C_{opt.}}$ в зависимости от параметров $P_L$ и $P_H$ (рис. 5.29).

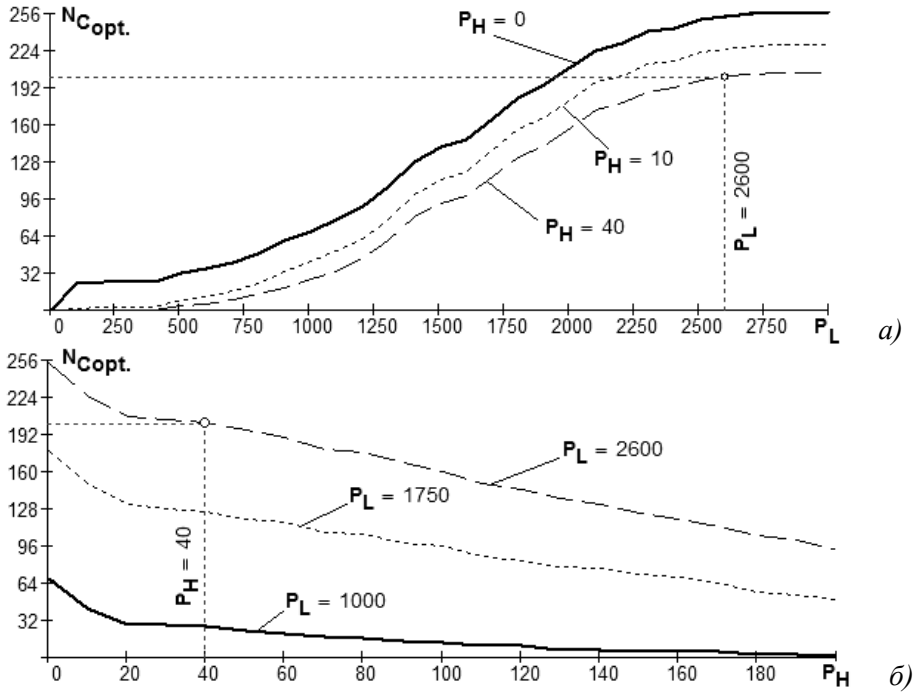



**Рис. 5.29. Зависимость количества пригодных для встраивания блоков от значений порогов $P_L$ и $P_H$: $P_L$ = *var*, $P_H$ = *const* (*а*); $P_L$ = *const*, $P_H$ = *var* (*б*)**

Циклом изменения параметра **c** в модуле (М.52) совершается выбор матрицы коэффициентов ДКП для **c**-го блока. Для выбранной матрицы подсчитывается сумма модулей коэффициентов, которые отвечают низкочастотному ($\Sigma_{LF}$) и высокочастотному ($\Sigma_{HF}$) участкам массива $\Omega'$ (см. рис. 5.27). Не учитывается только DC-коэффициент (условие ($\upsilon + \nu > 2$)). Если значения полученных сумм удовлетворяют поставленным требованиям, переменной **w** присваивается номер блока, признанного пригодным для встраивания, и цикл изменения **c** прерывается.

Проводится считывание коэффициентов массива $\Omega'$ (на котором произошла остановка), которые отвечают координатам ($\upsilon_1$; $\nu_1$), ($\upsilon_2$; $\nu_2$), ($\upsilon_3$; $\nu_3$). Если третий коэффициент не отвечает требованиям (5.22), проводится модификация двух коэффициентов из трех — собственно третьего коэффициента и коэффициента, который более всего отличается от третьего. Изменение происходит таким образом, чтобы разница между данными коэффициентами равнялась порогу различения **P**. Возможна и односторонняя модификация одного лишь третьего коэффициента (дополнительно изображено в листинге для модуля (М.52)), однако в этом случае нарушения структуры изображения будут более значительными.

Модифицированные значения коэффициентов заменяют соответствующие по координатам значения в массиве $\Omega'$, который, в свою очередь, присваивается **w**-му элементу результирующего массива $\Omega_M$.

**Шаг 4**

Используя модуль (М.53), проводим обратное дискретно-косинусное преобразование.

$$
\begin{aligned}
&C_M := \Big| \text{for } b \in 1..N_C \\
&\quad \Big| \text{for } x \in 0..N-1 \\
&\quad\quad \text{for } y \in 0..N-1 \\
&\quad\quad\quad C'M_{x+1,y+1} \leftarrow \left| \frac{1}{\sqrt{2\cdot N}} \cdot \sum_{\upsilon=0}^{N-1} \sum_{\nu=0}^{N-1} \left[ \zeta(\upsilon)\cdot\zeta(\nu)\cdot\left(\Omega M_b\right)_{\upsilon+1,\nu+1} \times \right.\right. \\
&\quad\quad\quad\quad \left.\left. \times \cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \right| \\
&\quad\quad C M_b \leftarrow \frac{C'M}{\max(C'M)}\cdot 255 \quad \text{if } \max(C'M) > 255 \\
&\quad\quad C M_b \leftarrow C'M \quad \text{otherwise} \\
&\quad C_M
\end{aligned}
\tag{М.53}
$$

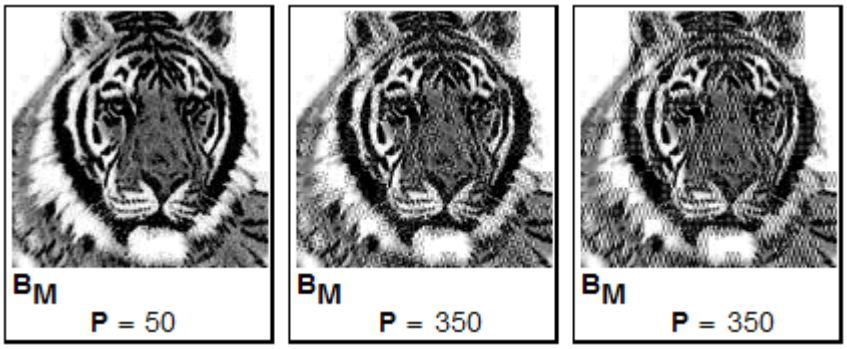
При этом, как и в предварительно рассмотренном методе, необходимо учесть возможность выхода значений интенсивностей пикселей за допустимые границы. Нормирование, аналогичное использованному в (М.50), нецелесообразно по причине того, что будет нормироваться весь массив контейнера и при извлечении возрастет вероятность ошибочного отбраковывания блоков, в которые передающей стороной было произведено встраивание бит информации (ошибка второго рода). Вследствие этого предлагается провести нормирование непосредственно результатов обратного ДКП отдельных блоков, а именно таких, у которых максимальное абсолют-

ное значение любого элемента превышает 255. В противном случае элементам блока контейнера возвращается только их абсолютное значение.

**Шаг 5**

Формирование на основе блоков общего изображения выполняется программным модулем (М.50), однако уже без нормирования значений элементов контейнера на завершающем этапе.

Графическое представление восстановленного массива синей составляющей при указанных выше начальных данных приведено на рис. 5.30*а*. На рис. 5.30*б* представлен вид, который приобретает данный массив в случае установления значения порога различения **P** := 350 при неизменных прочих значениях (модифицируются один из первых двух коэффициентов и третий). На рис. 5.30*в* изображен результат односторонней модификации третьего коэффициента ДКП (см. (М.52)) при неизменных прочих данных.




*а)* *б)* *в)*

**Рис. 5.30. Результаты встраивания сообщения M в массив канала синего при выборе оптимальных для встраивания блоков на основе разных значений порога P и при различных алгоритмах модификации коэффициентов ДКП**

Сравнение рис. 5.30 и рис. 5.28 приводит к очевидному выводу о преимуществе метода Бенгама и др. над методом Коха и Жао, поскольку первый позволяет эффективно отобрать именно те блоки изображения, встраивание в которые будет наименее заметным. Однако пропускная способность стеганосистем, построенных на основе рассматриваемого метода, будет уступать данному показателю систем, в основу работы которых заложен метод Коха и Жао.

**Шаг 6**

В процессе извлечения повторяется анализ блоков, аналогичный выполненному на этапе встраивания сообщения в контейнер.

Предварительно должны быть известными: алгоритм скрытия, массив-контейнер (**B***) и его размерность (**X***, **Y***), размерность сегментов (**N***) и координаты коэффициентов ДКП, которые использовались во время скрытия (или алгоритм их получения): ($\upsilon^*_1$; $\nu^*_1$), ($\upsilon^*_2$; $\nu^*_2$) и ($\upsilon^*_3$; $\nu^*_3$).

Проводится вычисление общего количества сегментов в изображении-контейнере — $N^*_C$.

Разбивка массива **B*** на сегменты $C^*_b$ выполняется программным модулем (М.45). К каждому сегменту применяется прямое ДКП (М.46) и (М.47), результатом чего является массив $\Omega^*$ коэффициентов ДКП сегментов $C^*_b$.

Извлечение скрытой информации совершается программным модулем (М.54), в основу которого положена проверка блока на его пригодность к встраиванию.

```
M* ≔ | i ← 1
     | j ← 1
     | w ← 0
     | for μ ∈ 1 .. N*C
     |   | w ← w + 1
     |   | for c ∈ w .. N*C
     |   |   | break if w > NC
     |   |   | Ω' ← Ω*c
     |   |   | ΣLF ← 0
     |   |   | v$ ← N* − 1
     |   |   | for υ ∈ 1 .. N* − 1
     |   |   |   for ν ∈ 1 .. v$
     |   |   |     | ΣLF ← ΣLF + |Ω'υ,ν| if (υ + ν) > 2
     |   |   |     | v$ ← v$ − 1 if ν = v$
     |   |   | ΣHF ← 0
     |   |   | v$ ← N*
     |   |   | for υ ∈ 3 .. N*
     |   |   |   for ν ∈ v$ .. N*
     |   |   |     | ΣHF ← ΣHF + |Ω'υ,ν|
     |   |   |     | v$ ← v$ − 1 if ν = v$
     |   |   | if ΣLF < P*L ∧ ΣHF > P*H
     |   |   |   | w ← c
     |   |   |   | break
     |   | ω1 ← Ω'υ*1,ν*1
     |   | ω2 ← Ω'υ*2,ν*2
     |   | ω3 ← Ω'υ*3,ν*3
     |   | ωmin ← min((ω1 ω2))
     |   | ωmax ← max((ω1 ω2))
     |   | m*i ← 0 if ω3 < ωmin
     |   | m*i ← 1 if ω3 > ωmax
     |   | i ← i + 1 if rows(m*) < 8
     |   | if rows(m*) = 8
     |   |   | i ← 1
     |   |   | M*j ← B2D(m*)
     |   |   | m* ← 0
     |   |   | j ← j + 1
     | vec2str(M*)
```

(М.54)

По результатам проверки в (M.54) делается вывод о том, был ли выбран рассматриваемый блок передающей стороной в качестве контейнера для бита сообщения, или же он был отбракован как не отвечающий требованиям пригодности. При положительном решении выполняется проверка соотношений (5.21), по результатам которой формируется вектор двоичных данных, преобразующийся в дальнейшем в символьную строку (аналогично тому, как это было реализовано в (M.51)).

Результаты вычисления показателей визуального искажения при **P** = 1 и **P** = 35 сведены в табл.5.4.

#### 5.3.3.3. Метод Xcy и By

Xcy (Chiou-Ting Hsu) и By (Ja-Ling Wu) [104] был предложен алгоритм встраивания цифрового водяного знака в массив коэффициентов ДКП блоков изображения-контейнера. Приведем основные положения, заложенные авторами в основу алгоритма.

Пусть $\boldsymbol{C}$ — полутоновое изображение размером $\boldsymbol{X}\times\boldsymbol{Y}$, а $\boldsymbol{W}$ — ЦВЗ, который представляет собой двоичное изображение размером $\boldsymbol{A}\times\boldsymbol{Z}$. В ЦВЗ пиксель может принимать значение или «1», или «0». Разумеется, что непосредственное наблюдение такого изображения невозможно, поскольку интенсивности 0 и 1 отвечают черному цвету (последняя — в некотором приближении). Изображение ЦВЗ можно создать черно-белым, а перед скрытием заменить интенсивность белых пикселей (255) на единицу, например, путем деления всего массива ЦВЗ на 255. При извлечении, наоборот, для визуального наблюдения массив ЦВЗ необходимо умножить на 255.

Поскольку, как будет показано в дальнейшем, во время встраивания ЦВЗ будет обрабатываться только среднечастотный диапазон сигнала-контейнера, необходимым предусловием является то, что ЦВЗ должен иметь меньший по сравнению с контейнером размер. Так, например, для контейнера, разбитого на блоки 8×8, при встраивании ЦВЗ оптимальным будет использование $64\cdot\boldsymbol{A}\cdot\boldsymbol{Z}/(\boldsymbol{X}\cdot\boldsymbol{Y})$ коэффициентов ДКП. Отношение $\boldsymbol{A}\cdot\boldsymbol{Z}/(\boldsymbol{X}\cdot\boldsymbol{Y})$ в данном случае определяет то количество информации, которое может быть встроено в выбранное в качестве контейнера изображение (в приведенном примере — до 64 коэффициентов блока 8×8). Для большей стойкости и скрытости результатов использования рассматриваемого стеганометода количество встроенной информации на практике пытаются уменьшить.

Изображение-контейнер $\boldsymbol{C}$ и ЦВЗ $\boldsymbol{W}$ представим как

$$\boldsymbol{C} = \{\boldsymbol{c}(\boldsymbol{x},\boldsymbol{y});\ 1 \le \boldsymbol{x} \le \boldsymbol{X};\ 1 \le \boldsymbol{y} \le \boldsymbol{Y}\}, \tag{5.23}$$

$$\boldsymbol{W} = \{\boldsymbol{w}(\boldsymbol{a},\boldsymbol{z});\ 1 \le \boldsymbol{a} \le \boldsymbol{A};\ 1 \le \boldsymbol{z} \le \boldsymbol{Z}\}, \tag{5.24}$$

где $\boldsymbol{c}(\boldsymbol{x},\boldsymbol{y}) \in \{0, ..., 2^{\boldsymbol{L}}-1\}$ — интенсивность пикселя $(\boldsymbol{x},\boldsymbol{y})$; $\boldsymbol{L}$ — количество бит, которое используется для квантования интенсивностей; $\boldsymbol{w}(\boldsymbol{a},\boldsymbol{z}) \in \{0, 1\}$ — двоичные значения пикселя $(\boldsymbol{a},\boldsymbol{z})$ ЦВЗ.

Контейнер $\boldsymbol{C}$ можно разбить на $\frac{\boldsymbol{X}}{8}\times\frac{\boldsymbol{Y}}{8}$ блоков размерностью 8×8. Для получения этого же количества блоков, ЦВЗ разбивается на блоки размерностью $\frac{8\cdot\boldsymbol{A}}{\boldsymbol{X}}\times\frac{8\cdot\boldsymbol{Z}}{\boldsymbol{Y}}$. Например, если $\boldsymbol{A} = \boldsymbol{X}/2$ и $\boldsymbol{Z} = \boldsymbol{Y}/2$, размерность блока ЦВЗ составит 4×4; если же $\boldsymbol{A} = \boldsymbol{X}/4$ и $\boldsymbol{Z} = \boldsymbol{Y}/4$, — 2×2 и т.д.

Для создания контейнера или ЦВЗ необходимой размерности, к последним могут быть добавлены дополнительные столбцы или строки.

#### Псевдослучайная перестановка пикселей ЦВЗ

В некотором приближении, каждый блок ЦВЗ встраивается в среднечастотные коэффициенты ДКП каждого блока контейнера путем использования блочного преобразования всего контейнера. Поэтому, вместо контейнера в целом, каждый блок ЦВЗ будет рассеян только лишь по соответствующему его блоку. При этом очевидно, что в случае отсутствия надлежащего регулирования пространственных связей ЦВЗ, обычное масштабирование контейнера может привести к разрушению ЦВЗ.

Для обеспечения стойкости к масштабированию, с целью изменения порядка ЦВЗ для рассредоточения его пространственных связей, авторами было предложено использовать быстрый метод генерации двумерного ПСЧ:

$$\boldsymbol{W}_{rnd} = \mathbf{permute}(\boldsymbol{W}); \tag{5.25}$$

$$\boldsymbol{W}_{rnd} = \{\boldsymbol{w}_{rnd}(\boldsymbol{a}, \boldsymbol{z}) = \boldsymbol{w}_{rnd}(\boldsymbol{a}', \boldsymbol{z}');\ 1 \le \boldsymbol{a} \le \boldsymbol{A};\ 1 \le \boldsymbol{z} \le \boldsymbol{Z}\},$$

где пиксель $(\boldsymbol{a}', \boldsymbol{z}')$ представляет собой переставленный в соответствии с псевдослучайной перестановкой (оператор **permute**) пиксель $(\boldsymbol{a}, \boldsymbol{z})$.

#### Перестановка блоков ЦВЗ, в зависимости от характеристик блоков контейнера

В соответствии с увеличением уровня скрытости, должны быть учтены характеристики контейнера (например, известно, что модификация высокочастотного диапазона или же участков с большей яркостью является менее заметной). Подобные, зависящие от контейнера, свойства могут быть использованы для перестановки псевдослучайно смешанного ЦВЗ для получения большего соответствия чувствительности ЗСЧ.

Авторы предлагают упорядочить блоки контейнера в соответствии с изменением дисперсий интенсивностей пикселей (например, по их уменьшению). В свою очередь, блоки ЦВЗ сортируются по количеству информации (то есть количеству значащих (единичных) пикселей). Вид сортировки блоков ЦВЗ (по возрастанию или по убыванию) должен отвечать аналогичной операции над блоками контейнера. Таким образом, каждому блоку контейнера отвечает свой блок ЦВЗ, то есть

$$\boldsymbol{W}_{sort} = \mathbf{permute}(\boldsymbol{W}_{rnd}). \tag{5.26}$$

На рис. 5.31 приведен пример сортировки и перестановки блоков.

| *Индекс блока контейнера* | *Значение дисперсии* |
|---|---|
| 7 | 67.1 |
| 2 | 61.2 |
| 3 | 53.0 |
| 6 | 41.9 |
| 1 | 32.3 |
| 5 | 14.5 |
| 4 | 7.4 |

| *Индекс блока ЦВЗ* | *Количество единиц* |
|---|---|
| 4 | 15 |
| 1 | 13 |
| 3 | 10 |
| 2 | 9 |
| 6 | 8 |
| 7 | 6 |
| 5 | 3 |

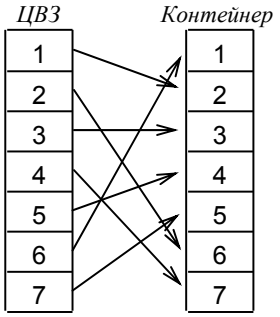



**Рис. 5.31. Пример перестановки блоков ЦВЗ в зависимости от характеристик блоков контейнера**

#### Преобразование блоков контейнера

Поскольку ДКП, используемое при JPEG-компрессии, оперирует с блоками размерностью 8×8, представляется целесообразным разбить контейнер $\boldsymbol{C}$ на блоки указанной размерности. К каждому блоку применяется операция прямого ДКП (оператор **FDCT**):

$$\boldsymbol{\Omega} = \mathbf{FDCT}(\boldsymbol{C}). \tag{5.27}$$

#### Выбор среднечастотных коэффициентов ДКП

Для того чтобы встроенный ЦВЗ визуально был незаметным и оставался при этом достаточно стойким к компрессии данных с потерями, очевидным компромиссом является его встраивание в диапазон средних частот контейнера. При этом для каждого блока 8×8 контейнера из возможных 64-х отбираются $64 \cdot \boldsymbol{A} \cdot \boldsymbol{Z} / (\boldsymbol{X} \cdot \boldsymbol{Y})$ коэффициентов ДКП, расположенных вдоль второстепенной диагонали матрицы ДКП. Отобранные коэффициенты для удобства последующих действий сворачиваются в уменьшенную матрицу размерностью $\frac{8 \cdot \boldsymbol{A}}{\boldsymbol{X}} \times \frac{8 \cdot \boldsymbol{Z}}{\boldsymbol{Y}}$ (оператор **reduce**):

$$\boldsymbol{\Omega}_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}). \tag{5.28}$$

В частности, если $\boldsymbol{A} = \boldsymbol{X}/2$ и $\boldsymbol{Z} = \boldsymbol{Y}/2$, во время встраивания ЦВЗ обрабатываются только 16 коэффициентов ДКП, а другие 48 остаются неизмененными. Рассмотренный выше процесс формирования массива СЧ-коэффициентов ДКП проиллюстрирован на рис. 5.32.

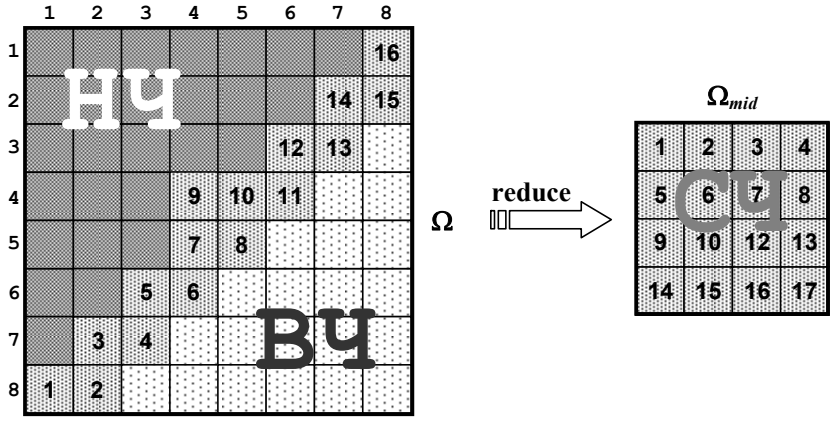



**Рис. 5.32. Конфигурация матрицы Ω коэффициентов ДКП и пример сведения СЧ-коэффициентов в отдельную матрицу $\boldsymbol{\Omega}_{mid}$**

#### Модификация коэффициентов ДКП

В результате предыдущих действий имеем: переставленный в псевдослучайном порядке и приведенный в соответствие к блоку контейнера блок ЦВЗ, а также приведенное частотное отображение контейнера (содержащее только СЧ-компоненты первичного изображения), — оба размерностью $\frac{8 \cdot \boldsymbol{A}}{\boldsymbol{X}} \times \frac{8 \cdot \boldsymbol{Z}}{\boldsymbol{Y}}$.

Элементы массива $\boldsymbol{\Omega}_{mid}$ могут быть модифицированы в соответствии с пикселями встроенного блока ЦВЗ.

По мнению авторов алгоритма, эффективным средством достижения незаметности ЦВЗ и стойкости стеганосистемы при низких коэффициентах JPEG-компрессии является встраивание каждого пикселя ЦВЗ путем модификации полярности между соответствующими пикселями соседних блоков. Однако при этом оговаривается, что такой метод не будет стойким к атакам при высоком ($\geq 6$) коэффициенте компрессии.

Предлагается рассмотреть технические аспекты проблемы встраивания при указанном подходе, а также усовершенствованный метод, являющийся более стойким к атакам компрессии.

#### Встраивание путем модификации отношения между значениями коэффициентов соседних блоков

Для расчета полярности выбранных СЧ-коэффициентов соседних блоков используется так называемая «остаточная» маска. На рис. 5.33 изображен пример такой маски, где каждый элемент (от «А» до «И») содержит в себе приведенное отображение коэффициентов ДКП контейнера $\Omega_{mid}$ некоторого блока, причем позиции «Д» отвечает текущее отображение ДКП.

Например, если А = Б = В = Е = Ж = З = И = *0*, Г = *–1*, а Д = *1*, то полярность будет представлять собой двоичный образ — массив нулей и единиц, указывающий на тот факт, что коэффициент ДКП текущей позиции данного блока отображения коэффициентов ДКП является б*о*льшим (полярность равна 1) или меньшим (полярность равна 0) по сравнению с коэффициентом на соответствующей позиции предыдущего блока. То есть для приведенного примера

| | | |
|---|---|---|
| А | Б | В |
| Г | Д | Е |
| Ж | З | И |

**Рис.5.33. Пример остаточной маски**

$$\boldsymbol{P} = \mathbf{polarity}(\Omega_{mid}) = \begin{cases} 1, \text{ при } \omega_{\mathbf{mid}_b}(\upsilon,\nu) > \omega_{\mathbf{mid}_{b-1}}(\upsilon,\nu); \\ 0, \text{ при } \omega_{\mathbf{mid}_b}(\upsilon,\nu) \leq \omega_{\mathbf{mid}_{b-1}}(\upsilon,\nu), \end{cases} \quad (5.29)$$

где $\omega_{\mathbf{mid}_b}(\upsilon,\nu)$ — среднечастотный коэффициент ДКП ***b***-го блока; **polarity** — оператор полярности.

При иных значениях элементов остаточной маски соответственно изменяется и выражение (5.29). При вычислении полярности сравнение значения коэффициента ДКП текущего блока со значениями соответствующих коэффициентов нескольких соседних блоков в большинстве случаев позволяет, кроме увеличения уровня защищенности стеганосистемы от взлома, получить меньшее искажение контейнера.

После получения отображений полярности ***P*** для всех блоков контейнера, проводится выявление тех коэффициентов ДКП, которые требуют модификации для скрытия отдельного пикселя псевдослучайно переставленного ЦВЗ. Поиск проводится в соответствии с остаточной маской путем изменения текущей полярности (оператор **XOR** или знак «⊕» — сложение по модулю 2):

$$\hat{\boldsymbol{P}} = \boldsymbol{XOR}\left(\boldsymbol{P}, \boldsymbol{W}_{sort}\right); \quad (5.30)$$

$$\hat{\boldsymbol{P}} = \{\hat{\boldsymbol{p}}(\upsilon,\nu);\ 1 \leq \upsilon \leq \frac{8 \cdot \boldsymbol{A}}{\boldsymbol{X}};\ 1 \leq \nu \leq \frac{8 \cdot \boldsymbol{Z}}{\boldsymbol{Y}}\},$$

где $\hat{\boldsymbol{p}}(\upsilon,\nu) = \begin{cases} 1 - \boldsymbol{p}(\upsilon,\nu), \text{ при } \boldsymbol{w}_{sort}(\upsilon,\nu) = 1; \\ \boldsymbol{p}(\upsilon,\nu), \text{ при } \boldsymbol{w}_{sort}(\upsilon,\nu) = 0. \end{cases} = \boldsymbol{p}(\upsilon,\nu) \oplus \boldsymbol{w}_{sort}(\upsilon,\nu).$

Далее, на основе массивов полярности $\hat{\boldsymbol{P}}$ для каждого блока контейнера формируют массив $\hat{\boldsymbol{\Omega}}_{mid}$ модифицированных СЧ-коэффициентов ДКП при условии, чтобы разница между $\boldsymbol{\Omega}_{mid}$ и $\hat{\boldsymbol{\Omega}}_{mid}$ была сведена к минимуму или же была меньшей за установленный порог $\eta$ (оператор **expand**):

$$\hat{\boldsymbol{\Omega}}_{mid} = \mathbf{expand}(\hat{\boldsymbol{P}}), \text{ при } \sum_{\upsilon,\nu}\left[w_{sort}(\upsilon,\nu) - \hat{w}_{sort}(\upsilon,\nu)\right]^2 < \eta. \qquad (5.31)$$

Например, задаваясь начальным коэффициентом $\hat{w}_{sort}(\upsilon_1,\nu_1) = w_{sort}(\upsilon_1,\nu_1)$, необходимо прибавлять/отнимать коэффициенты соседних блоков (в соответствии с остаточной маской) таким образом, чтобы, проведя впоследствии операцию, аналогичную (5.29), можно было получить соответствующую полярность $\hat{p}(\upsilon_1,\nu_1)$. Далее следует перейти к следующим коэффициентам, изменяя только те из них, которые не будут влиять на полярность предварительно обработанных коэффициентов.

Для того чтобы уменьшить деградацию изображения (как результат встраивания ЦВЗ), авторы метода предлагают вычислять полярность для абсолютных значений коэффициентов ДКП, что позволит гарантированно сохранить знак (плюс или минус) модифицированного коэффициента.

Кроме того, для увеличения стойкости стеганосистемы к JPEG-компрессии с потерями, должен быть учтен эффект квантования, который используется в технологии JPEG. На рис. 5.34*а* приведена таблица квантования яркости, предложенная стандартом JPEG, которая, зачастую, вызывает заметные искажения (так называемые «артефакты») изображения. На рис. 5.34*б* изображена другая таблица квантования, используемая в большинстве современных программ, работающих с JPEG. Видно, что значения при этом почти вдвое меньше соответствующих им в предыдущей таблице.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

*а)*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 6 | 5 | 8 | 12 | 20 | 26 | 31 |
| 6 | 6 | 7 | 10 | 13 | 29 | 30 | 28 |
| 7 | 7 | 8 | 12 | 20 | 29 | 35 | 28 |
| 7 | 9 | 11 | 15 | 26 | 44 | 40 | 31 |
| 9 | 11 | 19 | 28 | 34 | 55 | 52 | 39 |
| 12 | 18 | 28 | 32 | 41 | 52 | 57 | 46 |
| 25 | 32 | 39 | 44 | 52 | 61 | 60 | 51 |
| 36 | 47 | 48 | 49 | 56 | 50 | 52 | 50 |

*б)*

**Рис. 5.34. Примеры таблиц квантования яркости: стандартная JPEG (*а*) и Image Alchemy, Handmade Software Inc. (*б*)**

Основанная на таблице квантования полярность представляет собой результат вычисления разницы между квантовыми и впоследствии деквантовыми коэффициентами ДКП соответствующих блоков. Для тривиального случая, когда сравнение ведется с коэффициентами предшествующего блока (см. пример к рис. 5.33), формула (5.29) приобретает вид

$$P_b = \begin{cases} 1, \text{ при } \left[\dfrac{\left|\omega_{mid_b}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q_{mid}(\upsilon,\nu) > \left[\dfrac{\left|\omega_{mid_{b-1}}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q_{mid}(\upsilon,\nu); \\ 0, \text{ при } \left[\dfrac{\left|\omega_{mid_b}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q_{mid}(\upsilon,\nu) \le \left[\dfrac{\left|\omega_{mid_{b-1}}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q_{mid}(\upsilon,\nu), \end{cases} \quad (5.32)$$

где $Q_{mid}(\upsilon, \nu)$ — значение результата квантования для СЧ-коэффициента с координатами $(\upsilon, \nu)$; квадратные скобки указывают на то, что возвращается целая часть от результата деления.

При этом, в случае атаки квантования, предварительный учет эффекта последнего значительно увеличивает вероятность правильного распознания признаков пикселей при извлечении. Однако, поскольку квантование имеет тенденцию к сведению значений многих коэффициентов в нуль (что особенно характерно более высокочастотным коэффициентам), некоторая часть СЧ-коэффициентов ДКП также в результате будет равняться нулю. Кроме того, для сохранения установленной полярности и после проведения квантования, тем же значением должны быть модифицированы не только определенные СЧ-коэффициенты в текущем блоке, но и во всех соседних блоках в соответствии с маской остаточности.

#### Встраивание путем модификации отношения между значениями коэффициентов в пределах одного блока

Для преодоления описанных выше технических трудностей, Хсу и Ву предложили вместо сравнения с СЧ-коэффициентами ДКП соседних блоков использовать DC-коэффициент текущего блока. В этом случае,

$$P_b = \begin{cases} 1, \text{ при } \left[\dfrac{\left|\omega_{mid_b}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q(\upsilon,\nu) > \left[\dfrac{\omega_b(1,1)}{\psi\cdot Q(1,1)}\right]\cdot Q(1,1); \\ 0, \text{ при } \left[\dfrac{\left|\omega_{mid_b}(\upsilon,\nu)\right|}{Q_{mid}(\upsilon,\nu)}\right]\cdot Q(\upsilon,\nu) \le \left[\dfrac{\omega_b(1,1)}{\psi\cdot Q(1,1)}\right]\cdot Q(1,1), \end{cases} \quad (5.33)$$

где $\psi$ — масштабный коэффициент; $Q(1,1)$ — значение квантования для DC-коэффициента.

#### Обратное преобразование блоков контейнера

Модифицированные матрицы СЧ-коэффициентов ($\hat{\Omega}_{mid}$) отображаются в общие матрицы коэффициентов ДКП ($\Omega$) (оператор **put**):

$$\hat{\Omega} = \mathbf{put}(\hat{\Omega}_{mid}). \quad (5.34)$$

К результату проведенного объединения применяется обратное ДКП (оператор **IDCT**):

$$\hat{C} = \mathbf{IDCT}(\hat{\Omega}). \quad (5.35)$$

#### Извлечение ЦВЗ из контейнера

Процесс извлечения требует наличия оригинального изображения-контейнера, изображения со встроенным ЦВЗ, а также изображения, выступающего в роли ЦВЗ.

Оба изображения (оригинальное — $\boldsymbol{C}$, и исследуемое на наличие встроенного ЦВЗ — $\boldsymbol{C}^*$) подвергаются прямому ДКП: $\boldsymbol{\Omega} = \mathbf{FDCT}(\boldsymbol{C})$, $\boldsymbol{\Omega}^* = \mathbf{FDCT}(\boldsymbol{C}^*)$.

Из полученных массивов коэффициентов ДКП проводится выделение матриц СЧ-коэффициентов, которые, в свою очередь, используются для получения шаблонов полярности:

$$\boldsymbol{\Omega}_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}), \quad \boldsymbol{P} = \mathbf{polarity}(\boldsymbol{\Omega}_{mid});$$
$$\boldsymbol{\Omega}^*_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}^*), \quad \boldsymbol{P}^* = \mathbf{polarity}(\boldsymbol{\Omega}^*_{mid}).$$

Применяя к полученным массивам полярностей операцию сложения по модулю 2, получают двоичные данные (пока еще переставленные в пространстве и псевдослучайно смешанные):

$$\boldsymbol{W}^*_{sort} = \mathbf{XOR}(\boldsymbol{P}, \boldsymbol{P}^*), \tag{5.36}$$

где $\boldsymbol{w}^*_{sort}(\upsilon, \nu) = \boldsymbol{p}(\upsilon, \nu) \oplus \boldsymbol{p}^*(\upsilon, \nu)$.

Выполняется обратная пространственная перестановка блоков полученных данных (оператор **re-permute**). Индексы соответствующих пар блоков и извлекаемых данных могут быть получены либо путем их считывания из предварительно сохраненных в файле на этапе встраивания ЦВЗ, или же непосредственно при встраивании путем аналогичных действий над изображением-оригиналом и изображением-ЦВЗ:

$$\boldsymbol{W}^*_{rnd} = \mathbf{re\text{-}permute}(\boldsymbol{W}^*_{sort}).$$

Аналогично проводится обратная псевдослучайная перестановка данных в полученном массиве: $\boldsymbol{W}^* = \mathbf{re\text{-}permute}(\boldsymbol{W}^*_{rnd})$.

В рассмотренном алгоритме можно выделить три особенности, которые можно использовать в качестве секретного ключа:

1. начальное число генератора ПСЧ, которое будет определять первый элемент псевдослучайной перестановки (любое целое число из диапазона $[1, \boldsymbol{A}\cdot\boldsymbol{Z}–1]$);
2. выбор СЧ-коэффициентов ДКП (необходимо выбрать $64\cdot\boldsymbol{A}\cdot\boldsymbol{Z}/(\boldsymbol{X}\cdot\boldsymbol{Y})$ коэффициенты из 64-х для каждого блока, следовательно, за каждым блоком можно закрепить свой набор коэффициентов);
3. алгоритм сведения (отображения) выбранных СЧ-коэффициентов в отдельную матрицу (на рис. 5.32 показан только один из возможных способов такого отображения).

Рассмотрим пример реализации метода Хсу и Ву в системе MathCAD.

**Шаг 1**

Пусть изображение-контейнер и изображение-ЦВЗ представляют собой графические файлы C.bmp и W.bmp соответственно (рис. 5.35):

**C** := **READBMP**("C.bmp");
**W** := **READBMP**("W.bmp").

При этом соответствующие характеристики указанных изображений составляют:

**X** := **rows**(**C**), **X** = 128;
**Y** := **cols**(**C**), **Y** = 128;
**A** := **rows**(**W**), **A** = 64;
**Z** := **cols**(**W**), **Z** = 64.


**Рис. 5.35. Изображение-контейнер и внедряемый в него ЦВЗ**

**Шаг 2**

Проводим нормирование массива ЦВЗ:

$$W := \overrightarrow{round(W \div max(W))}.$$

Элементы исходного массива **W** при этом принимают значение 0 или 1.

**Шаг 3**

Размерность блоков, на которые разбивается контейнер, принимаем равной $N := 8$. Количество полученных в этом случае блоков: $\aleph_C := X \cdot Y \div N^2$, $\aleph_C = 256$.

Размерность, которую должны иметь блоки ЦВЗ для получения их количества $\aleph_W = \aleph_C$, составляет $n := A \cdot N \div X$, $n = 4$.

Выполняем проверку количества получаемых блоков ЦВЗ: $\aleph_W := A \cdot Z \div n^2$, $\aleph_W = 256$ блоков.

**Шаг 4**

Разбитие контейнера **C** на $\aleph_C$ блоков размерностью $N \times N$ выполняем при помощи программного модуля (М.55).

```
БC := | c1 ← 1
      | c2 ← N
      | for b ∈ 1 .. ℵC
      |     | r1 ← mod[N·(b − 1) + 1 , X]
      |     | r2 ← r1 + N − 1
      |     | БCb ← submatrix(C , r1 , r2 , c1 , c2)
      |     | if r2 = X
      |     |     | c1 ← c1 + N
      |     |     | c2 ← c2 + N
      | БC
```

(М.55)

**Шаг 5**

Псевдослучайную перестановку элементов ЦВЗ проведем в том порядке, который был предложен Хсу и Ву. Во-первых, проведем индексацию каждого пикселя ЦВЗ (от 1 до $A \cdot Z = 4096$), для чего просто развернем массив ЦВЗ в вектор (программный модуль (М.56)).

```
Wvec := | Wvec ← W⟨1⟩
        | for z ∈ 2 .. Z
        |     Wvec ← stack(Wvec , W⟨z⟩)
```

(М.56)

Во-вторых, полученные индексы расставим в произвольном (псевдослучайном) порядке, для чего используем ***линейный регистр сдвига с обратной связью*** (ЛРСОС или LFSR — Linear Feedback Shift Register).

Как известно, ЛРЗЗЗ состоит из двух частей: собственно регистра сдвига и функции обратной связи (рис. 5.36) [65]. Регистр сдвига представляет собой последовательность битов (разрядов) ***R***, количество которых ***d*** определяется длиной регистра сдвига. Обратная связь представляет собой сумму по модулю 2 определенных битов регистра (эти биты называются ***соответствующей последовательностью***).

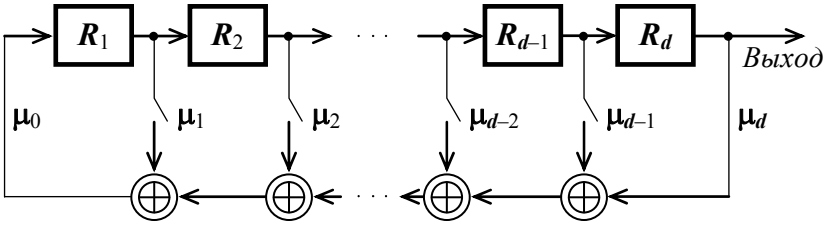



**Рис. 5.36. Обобщенный линейный регистр сдвига с обратной связью**

Теоретически, ***d***-битовый ЛРСОС может пребывать в одном из $2^d$–1 внутренних состояний, то есть может генерировать ПСП с периодом в $T = 2^d$–1 бит. Все $T$ внутренних состояний регистр пройдет только при определенных отводных последовательностях. Такие ЛРСОС имеют максимальный период, а полученный при этом результат называют ***M-последовательностью***.

На рис. 5.36 значение $\mu_i$ ($i$ = 0, 1, ..., $d$) является весовым коэффициентами полинома $\rho(x)$ степени $d$, ассоциированного с последовательностью:

$$\rho(x) = \mu_0 \cdot x^0 + \mu_1 \cdot x^1 + \mu_2 \cdot x^2 + ... + \mu_{d-2} \cdot x^{d-2} + \mu_{d-1} \cdot x^{d-1} + \mu_d \cdot x^d.$$

Если $\mu_i = 1$, то соответствующий ключ замкнут. В случае $\mu_i = 0$ — разомкнут.

Неудачное включение сумматоров в цепь обратной связи может привести к получению ПСП, период повторения которой будет меньшим максимально возможного при имеющейся разрядности регистра. Для того чтобы конкретный ЛРСОС имел максимальный период, полином $\rho(x)$ должен быть примитивным по модулю 2 (то есть не раскладываться на произведение двоичных полиномов меньшей степени). При этом коэффициенты $\mu_0$ и $\mu_d$ всегда равняются 1, поскольку, в случае $\mu_0 = 0$, полином $\rho(x)$ делится на $x$ и не является примитивным; в случае $\mu_d = 0$, даже если полином и примитивный, его степень меньше $d$. Другие коэффициенты обратного полинома и будут определять схему формирователя ПСП.

В нашем случае, для перестановки чисел в диапазоне от 1 до **A**·**Z** необходимым и достаточным является количество разрядов регистра **d** := **log**(**A**·**Z**, 2), **d** = 12. При этом период повторения ПСП составит **A**·**Z** – 1 = 4095.

Для **d**-разрядного регистра в качестве примитивного полинома по модулю 2 выберем следующий: $\rho(x) = 1 + x + x^2 + x^8 + x^{12}$. Этот и некоторые другие возможные виды примитивных полиномов степени **d** = 12 приведены в табл. 5.2[8].

**Таблица 5.2.**

**Примеры примитивных по модулю 2 полиномов степени d = 12**

| $x^0$ | $x^1$ | $x^2$ | $x^3$ | $x^4$ | $x^5$ | $x^6$ | $x^7$ | $x^8$ | $x^9$ | $x^{10}$ | $x^{11}$ | $x^{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

[8] Для поиска примитивных полиномов в конечных полях (полях Галуа) **GF**($2^{\mathbf{d}}$) можно воспользоваться функцией MatLAB **gfprimfd**(**d**, **p**), где **d** — положительное целое число; **p** — параметр, который можно задать как 'max', 'min' (при этом возвращается один примитивный полином, имеющий, соответственно, максимально и минимально возможное число ненулевых коэффициентов) и 'all' (возвращаются все примитивные полиномы), а также в виде положительного целого числа (возвращаются полиномы, имеющие ровно **p** ненулевых коэффициентов).

ЛРСОС, имеющий **d** разрядов, реализует программный модуль (М.57), в котором аргумент **s** определяет собой начальное состояние регистра (в десятичном представлении) — произвольное целое число в пределах от 1 до **A·Z**–1.

```
                                  x^1 x^2 x^3                                  x^d
Vrnd(s) := | μ ← (1  1  0  0  0  0  0  1  0  0  0  1)^T
           | for i ∈ 1 .. 2^d − 1
           |   if i = 1
           |   |  R_dec_i ← s
           |   |  R_bin ← D2B(R_dec_i)
           |   if i ≥ 2
           |   |  bit ← 0
           |   |  for j ∈ 1 .. d
           |   |     bit ← R_bin_j ⊕ bit  if  μ_j = 1
           |   |  R ← R_bin
           |   |  for j ∈ 1 .. d
           |   |     R_bin_j ← R_(j−1)  if  j > 1
           |   |     R_bin_j ← bit  if  j = 1
           |   |  R_dec_i ← B2D(R_bin)
           | r_2d ← 2^d
           | r
```

(М.57)

В начале модуля задается вектор весовых коэффициентов примитивного полинома $\rho(x)$ для элементов отводной последовательности (для наглядности данный вектор изображен в виде матрицы-строки с последующим транспонированием).

Циклом изменения **i** проводится изменение состояния регистра. Каждое **i**-ое состояние из двоичного формата конвертируется в десятичный и сохраняется как значение соответствующего элемента вектора $\mathbf{R_{dec}}$. Поскольку период последовательности, генерируемой данным регистром, равняется $2^d - 1$, а псевдослучайная перестановка будет применяться к вектору, количество элементов которого равняется $\mathbf{A{\cdot}Z} = 2^d$, в конце модуля к сформированному вектору $\mathbf{R_{dec}}$ дописывается еще один элемент, значение которого учитывает верхний граничный индекс элементов вектора $\mathbf{W_{vec}}$ ($2^d$).

Получение массива **Vrnd** индексов элементов вектора $\mathbf{W_{vec}}$, расставленных в псевдослучайном порядке, позволяет в дальнейшем провести генерирование пар координат (по строкам и столбцам) каждого пикселя путем преобразования последовательности ПСЧ в двумерную последовательность. Это, в свою очередь, делает возможным после псевдослучайного выбора элемента из вектора $\mathbf{W_{vec}}$, поместить его значение в обусловленный сгенерированной парой координат элемент массива, размерность которого идентична размерности ЦВЗ.

Вышеприведенная процедура реализована в программном модуле (М.58), в котором для каждого элемента М-последовательности вычисляются индексы **a** и **z** элемента массива $\mathbf{W_{rnd}}$, в который заносится текущий элемент вектора $\mathbf{W_{vec}}$. Функция **trunc**(*x*) возвращает целую часть от аргумента *x*, отбрасывая его мантиссу; функция **mod**(*k*, *m*) возвращает остаток от деления *k* на *m*. Прибавлением единицы учтена возможность возвращения отмеченными функциями нулевого результата.

$$W_{rnd} := \left| \begin{array}{l} \Im \leftarrow Vrnd(s) \\ \text{for } i \in 1..A \cdot Z \\ \quad \left| \begin{array}{l} a \leftarrow trunc\left(\dfrac{\Im_i - 1}{A}\right) + 1 \\ z \leftarrow mod(\Im_i, Z) + 1 \\ W_{rnd_{a,z}} \leftarrow W_{vec_i} \end{array} \right. \\ W_{rnd} \end{array} \right. \qquad \text{(M.58)}$$

Результат выполнения модуля (M.58) при $s := 12$ приведен на рис. 5.37.

**Шаг 6**


**Рис. 5.37. Результат псевдослучайной перестановки элементов ЦВЗ**

Модуль разбития массива ЦВЗ на $\aleph_W$ блоков размерностью $n \times n$ по своему строению аналогичен модулю (M.55). Отличия заключаются в следующем:

- переменная, которой присваивается результат выполнения модуля (как, собственно, и соответствующая переменная в теле модуля), обозначается как $Б_W$ (вместо $Б_C$);
- выделение блоков проводится из массива $W_{rnd}$ (вместо $C$);
- вместо размерности массива $N$ используется размерность $n$;
- соответственно, изменяется и предельное значение индекса строки (вместо $X$ — значение $A$).

Общее количество блоков, на которое разбивался контейнер ($\aleph_C$), с учетом такого же их количества в ЦВЗ, можно не изменять.

**Шаг 7**

С помощью программных модулей (M.59) и (M.60) формируем таблицы результатов сортировки блоков контейнера (по значениям стандартного отклонения элементов блоков) и блоков ЦВЗ (по количеству значащих элементов).

$$T_C := \left| \begin{array}{l} \text{for } b \in 1..\aleph_C \\ \quad \left| \begin{array}{l} T_{b,1} \leftarrow b \\ \sigma \leftarrow \sqrt{\dfrac{1}{N^2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\left(Б_{C_b}\right)_{i,j}\right]^2 - \left[\dfrac{1}{N^2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \left(Б_{C_b}\right)_{i,j}\right]^2} \\ T_{b,2} \leftarrow \sigma \end{array} \right. \\ reverse(csort(T, 2)) \end{array} \right. \qquad \text{(M.59)}$$

В первый столбец таблиц характеристик блоков контейнера ($T_C$) и ЦВЗ ($T_W$) вносятся порядковые индексы исследуемых блоков. Во второй — результат вычисления, соответственно, стандартного отклонения ($\sigma$) и количества значащих (единичных) элементов ($\Sigma 1$). После формирования таблиц, последние сортируются по значениям второго столбца (функция $csort(T,2)$).

```
T_W := | for b ∈ 1 .. ℵ_W
       |   T_{b,1} ← b
       |   Σ1 ← 0
       |   for υ ∈ 1 .. n
       |     for ν ∈ 1 .. n
       |       Σ1 ← Σ1 + (Б_{W_b})_{υ,ν}
       |   T_{b,2} ← Σ1
       | reverse(csort(T, 2))
```

(М.60)

Фрагмент результата сортировки для выбранных контейнера и ЦВЗ приведен в табл.5.3.

**Таблица 5.3.**

**Пример сортировки блоков контейнера и ЦВЗ**

| № п/п | Индексы блоков контейнера | Значения станд. отклонения | Индексы блоков ЦВЗ | Количество значащих элементов |
|---|---|---|---|---|
| 1 | 232 | 66,676 | 84 | 15 |
| 2 | 54 | 65,551 | 183 | 15 |
| 3 | 55 | 65,236 | 185 | 15 |
| ... | ... | ... | ... | ... |
| 64 | 45 | 44,565 | 152 | 12 |
| 65 | 190 | 44,394 | 153 | 12 |
| 66 | 98 | 44,318 | 154 | 12 |
| ... | ... | ... | ... | ... |
| 128 | 81 | 33,562 | 169 | 11 |
| 129 | 79 | 33,462 | 117 | 11 |
| 130 | 209 | 33,287 | 245 | 11 |
| ... | ... | ... | ... | ... |
| 192 | 175 | 22,564 | 48 | 10 |
| 193 | 29 | 22,396 | 193 | 10 |
| 194 | 239 | 21,151 | 194 | 10 |
| ... | ... | ... | ... | ... |
| 254 | 3 | 0 | 138 | 7 |
| 255 | 1 | 0 | 50 | 7 |
| 256 | 241 | 0 | 46 | 6 |

Путем выделения первых столбцов из массивов $T_C$ и $T_W$, сопоставляем индексы блоков контейнера с индексами блоков ЦВЗ:

$$T_\Sigma := \text{augment}\left(T_C^{\langle 1 \rangle}, T_W^{\langle 1 \rangle}\right).$$

**Шаг 8**

В соответствии с полученным массивом приведенных в соответствие друг с другом индексов, проводим перестановку блоков ЦВЗ в порядке, отвечающем данному соответствию. Реализацию данного этапа осуществляет модуль (М.61).

$$W_{sort} := \left| \begin{array}{l} \text{for } b \in 1 .. \aleph_C \\ \quad \left| \begin{array}{l} i1 \leftarrow T_{\Sigma_{b,1}} \\ i2 \leftarrow T_{\Sigma_{b,2}} \\ W_{sort_{i1}} \leftarrow Б_{W_{i2}} \end{array} \right. \\ W_{sort} \end{array} \right. \qquad \text{(M.61)}$$

В качестве примера приведем результат выполнения программного модуля (М.61) для первой строки табл. 5.3:

$$Б_{W_{84}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}; \qquad W_{sort_{232}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

**Шаг 9**

Используя программные модули (М.46,47) (при этом следует предельное количество блоков контейнера в цикле **for** заменить с $N_C$ на $\aleph_C$, а вместо $C_b$ под знаком суммы следует использовать $Б_{C_b}$), выполняем прямое ДКП блоков контейнера.

Из полученных массивов коэффициентов ДКП $\Omega_b$, которые имеют размерность $N \times N$, извлекаем только среднечастотные коэффициенты (см. рис. 5.32), параллельно сворачивая их в массив $n \times n$. Перед началом проведения операции извлечения формируется массив координат выделяемых СЧ-коэффициентов (М.62):

$$\Theta := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 14 & 15 \\ 0 & 0 & 0 & 0 & 0 & 12 & 13 & 0 \\ 0 & 0 & 0 & 9 & 10 & 11 & 0 & 0 \\ 0 & 0 & 0 & 7 & 8 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\theta := \left| \begin{array}{l} \text{for } i \in 1 .. N \\ \quad \text{for } j \in 1 .. N \\ \quad\quad \text{if } \Theta_{i,j} > 0 \\ \quad\quad\quad \left| \begin{array}{l} \theta_{\Theta_{i,j},1} \leftarrow i \\ \theta_{\Theta_{i,j},2} \leftarrow j \end{array} \right. \\ \theta \end{array} \right. \qquad \text{(M.62)}$$

В данном случае, массив $\theta$ будет содержать 16 строк, элементы каждой из которых несут информацию об индексах строки и столбца соответствующего СЧ-коэффициента в массиве $\Theta$ (рис.5.38).

$\theta^T =$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ⇐ порядковые № СЧ-коэф. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 8 | 7 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 1 | ⇐ индекс строки массива $\Theta$ |
| 2 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 4 | 5 | 6 | 6 | 7 | 7 | 8 | 8 | ⇐ индекс столбца массива $\Theta$ |

**Рис. 5.38. Таблица координат СЧ-коэффициентов ДКП**

С помощью программного модуля (М.63), в соответствии с таблицей рис. 5.38, для каждого блока $b$ выполняется формирование матрицы $\Omega_{mid}$ выбранных для модификации коэффициентов. Результат извлечения проиллюстрирован на рис. 5.39.

```
Ωmid := | for b ∈ 1 .. ℵC
        |   q ← 1
        |   for υ ∈ 1 .. n
        |     for ν ∈ 1 .. n
        |       Ω'mid_{υ,ν} ← (Ωb)_{(θq,1),(θq,2)}
        |       q ← q + 1
        |   Ωmid_b ← Ω'mid
        | Ωmid
```

(M.63)

$\Omega_{232}$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 833.62 | –494.34 | 109.89 | 99.26 | –72.62 | –12.92 | 28.49 | **–10.5** |
| –37.49 | 18.15 | 27.92 | –25.43 | –4.7 | 12.04 | **0.32** | **–14.48** |
| –48.45 | 30.19 | 31.45 | –33.88 | –11.36 | **29.37** | **0.62** | –10.97 |
| 7.72 | 1 | –10.55 | **–0.22** | **21.76** | **–0.45** | –11.96 | 16.57 |
| –6.12 | 1.32 | 11.16 | **–14.34** | **–0.37** | 1.08 | –0.16 | 0.65 |
| 0.01 | 0.16 | **14.31** | **–0.02** | –13.12 | 0.63 | 0.15 | –0.04 |
| –0.36 | **–0.35** | **0.62** | –0.55 | –0.3 | –0.37 | –0.2 | –0.54 |
| **–0.29** | **–0.06** | 0.55 | 0 | 0.85 | 0.84 | 0.15 | –0.06 |

$\Omega_{mid_{232}}$ =

| | | | |
|---|---|---|---|
| –0.29 | –0.06 | –0.35 | 0.62 |
| 14.31 | –0.02 | –14.34 | –0.37 |
| –0.22 | 21.76 | –0.45 | 29.37 |
| 0.62 | 0.32 | –14.48 | –10.5 |

**Рис. 5.39. Пример извлечения СЧ-коэффициентов из массива Ω для 232-го блока контейнера**

**Шаг 10**

Проводим вычисление массива полярностей блоков контейнера. При этом в основу программного модуля (М.64) положена реализация выражения (5.29). Матрица СЧ-коэффициентов 1-го блока сравнивается с матрицей последнего ($\aleph_C$-го) блока. Для создания более стойкого встраивания ЦВЗ данный модуль можно модифицировать в соответствии с (5.33) путем внесения соответствующей замены.

```
P := | for b ∈ 1 .. ℵC
     |   Δ ← Ωmid_b − Ωmid_{ℵC}      if b = 1
     |   Δ ← Ωmid_b − Ωmid_{b−1}      if b > 1
     |   Δ ← ( trunc( |Ωmid_b| / Qmid ) · Qmid ) − trunc( (Ωb)_{1,1} / (ψ · Q_{1,1}) ) · Q_{1,1}
     |   for υ ∈ 1 .. n
     |     for ν ∈ 1 .. n
     |       P'_{υ,ν} ← 1   if Δ_{υ,ν} > 0
     |       P'_{υ,ν} ← 0   if Δ_{υ,ν} ≤ 0
     |   P_b ← P'
     | P
```

(5.33)

(M.64)

**Шаг 11**

В соответствии с выражением (5.30) проводим изменение текущего массива полярности согласно значениям элементов переставленного ЦВЗ — модуль (М.65).

$$\begin{aligned}&\mathbf{P\hat{}} \leftarrow \Big|\ \text{for}\ \ b \in 1 .. \aleph_C \\ &\qquad \mathbf{P\hat{}}_b \leftarrow \overrightarrow{\left(P_b \oplus W_{sort_b}\right)} \\ &\mathbf{P\hat{}}\end{aligned} \tag{М.65}$$

**Шаг 12**

На основании текущих матриц СЧ-коэффициентов ($\Omega_{mid}$) формируем новые ($\Omega\hat{}_{mid}$), причем претерпевают изменения те элементы первичной матрицы, по координатам которых выполняется неравенство вида $P\hat{}_{\upsilon,\nu} \neq P_{\upsilon,\nu}$ (программный модуль (М.66)).

$$\begin{aligned}&\Omega\hat{}_{mid} \leftarrow \Big|\ \text{for}\ \ b \in 1 .. \aleph_C \\ &\qquad \Omega\hat{}'_{mid} \leftarrow \Omega_{mid_b} \\ &\qquad \text{for}\ \ \upsilon \in 1 .. n \\ &\qquad\quad \text{for}\ \ \nu \in 1 .. n \\ &\qquad\qquad \text{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} \neq \left(P_b\right)_{\upsilon,\nu} \\ &\qquad\qquad \left\{\begin{array}{l}\text{изменить при } P - \text{по (5.33)} \\ \text{if}\ \left(P\hat{}_b\right)_{\upsilon,\nu} = 0 \\ \quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{\aleph_C}}\right)_{\upsilon,\nu} - \eta \ \text{if}\ b = 1 \\ \quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{b-1}}\right)_{\upsilon,\nu} - \eta \ \text{if}\ b > 1 \\ \text{if}\ \left(P\hat{}_b\right)_{\upsilon,\nu} = 1 \\ \quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{\aleph_C}}\right)_{\upsilon,\nu} + \eta \ \text{if}\ b = 1 \\ \quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{b-1}}\right)_{\upsilon,\nu} + \eta \ \text{if}\ b > 1\end{array}\right. \\ &\qquad \Omega\hat{}_{mid_b} \leftarrow \Omega\hat{}'_{mid} \\ &\Omega\hat{}_{mid}\end{aligned} \tag{М.66 \textit{а}}$$

$$\begin{aligned}&\Big|\ \text{while}\ \ \text{trunc}\left(\frac{\left|\Omega\hat{}'_{mid_{\upsilon,\nu}}\right|}{Q_{mid_{\upsilon,\nu}}}\right)\cdot Q_{mid_{\upsilon,\nu}} - \text{trunc}\left[\frac{\left(\Omega_b\right)_{1,1}}{\psi\cdot Q_{1,1}}\right]\cdot Q_{1,1} > 0 \quad \text{if}\ \left(P\hat{}_b\right)_{\upsilon,\nu} = 0 \\ &\quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} - \eta \ \text{if}\ \Omega\hat{}'_{mid_{\upsilon,\nu}} > 0 \\ &\quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} + \eta \ \text{if}\ \Omega\hat{}'_{mid_{\upsilon,\nu}} < 0 \\ &\Big|\ \text{while}\ \ \text{trunc}\left(\frac{\left|\Omega\hat{}'_{mid_{\upsilon,\nu}}\right|}{Q_{mid_{\upsilon,\nu}}}\right)\cdot Q_{mid_{\upsilon,\nu}} - \text{trunc}\left[\frac{\left(\Omega_b\right)_{1,1}}{\psi\cdot Q_{1,1}}\right]\cdot Q_{1,1} \leq 0 \quad \text{if}\ \left(P\hat{}_b\right)_{\upsilon,\nu} = 1 \\ &\quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} + \eta \ \text{if}\ \Omega\hat{}'_{mid_{\upsilon,\nu}} \geq 0 \\ &\quad \Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} - \eta \ \text{if}\ \Omega\hat{}'_{mid_{\upsilon,\nu}} < 0\end{aligned} \tag{М.66 \textit{б}}$$

Модификация проводится таким образом, чтобы при выполнении программного модуля (М.64) можно было получить массив **P***, идентичный массиву **P^**, в том случае, если при вычислении параметра **Δ** в качестве уменьшаемого будет выступать элемент матрицы $\Omega^{\wedge}_{mid}$. В случае же модификации отношения между значениями коэффициентов в пределах одного блока (то есть полярность **P** предварительно определялась по формуле (5.33)), в модуле (М.66 *а*) выражение, вычисляемое при выполнении условия $P^{\wedge}_{\upsilon,\nu} \neq P_{\upsilon,\nu}$, необходимо заменить программным блоком (М.66 *б*).

**Шаг 13**

Модифицированные для каждого блока матрицы СЧ-коэффициентов ($\Omega_{mid}$) отображаются в общие матрицы коэффициентов ДКП ($\Omega^{\wedge}$) — программный модуль (М.67). При этом также используется таблица координат среднечастотных коэффициентов ДКП (см. (М.62) и рис. 5.38).

$$\begin{array}{l}\Omega^{\wedge} := \left|\begin{array}{l}\text{for } b \in 1..\aleph_C \\ \quad \left|\begin{array}{l}\Omega^{\wedge}{}' \leftarrow \Omega_b \\ q \leftarrow 1 \\ \text{for } \upsilon \in 1..n \\ \quad \text{for } \nu \in 1..n \\ \quad\quad \left|\begin{array}{l}\Omega^{\wedge}{}'_{(\theta_{q,1}),(\theta_{q,2})} \leftarrow \left(\Omega^{\wedge}mid_b\right)_{\upsilon,\nu} \\ q \leftarrow q+1\end{array}\right. \\ \Omega^{\wedge}_b \leftarrow \Omega^{\wedge}{}'\end{array}\right. \\ \Omega^{\wedge}\end{array}\right. \end{array} \qquad \text{(М.67)}$$

**Шаг 14**

После объединения, к модифицированным матрицам $\Omega^{\wedge}$ необходимо применить операцию обратного ДКП (модуль (М.68)) и сформировать на основе полученных блоков общий массив изображения-контейнера (модуль (М.69)).

$$\begin{array}{l}Б^{\wedge} := \left|\begin{array}{l}\text{for } b \in 1..\aleph_C \\ \quad \left|\begin{array}{l}\text{for } x \in 0..N-1 \\ \quad \text{for } y \in 0..N-1 \\ \quad\quad Б^{\wedge}{}'_{x+1,y+1} \leftarrow \frac{1}{\sqrt{2\cdot N}} \cdot \sum_{\upsilon=0}^{N-1}\sum_{\nu=0}^{N-1}\left[\zeta(\upsilon)\cdot\zeta(\nu)\cdot\left(\Omega^{\wedge}_b\right)_{\upsilon+1,\nu+1}\times\right. \\ \quad\quad\quad \left.\times\cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \\ \quad Б^{\wedge}_b \leftarrow Б^{\wedge}{}'\end{array}\right. \\ Б^{\wedge}\end{array}\right. \end{array} \qquad \text{(М.68)}$$

Полученное при этом изображение с встроенным ЦВЗ при определенных обстоятельствах может существенно потерять в яркости, что вызвано несколькими причинами: во-первых, для упрощения программных модулей не была проведена оптимизация встраивания по формуле (5.31); во-вторых, соседние блоки контейнера могут иметь достаточно разные значения интенсивностей и, соответственно, СЧ-коэффициентов ДКП, что вызывает необходимость при построении алгоритма по

(5.29) существенно изменять значения этих коэффициентов для удовлетворения поставленных условий. В комплексе эти две причины вызывают появление пикселей контейнера, яркость которых после проведения обратного ДКП, выходит за пределы диапазона [0, 255]. Последнее устраняется путем нормирования значений элементов массива **C^** в конце модуля (М.69), которое и вызывает снижение общей яркости изображения.

```
C^:= | C^ ← Б^1
     | for b ∈ 2 .. X/N
     |     C^ ← stack(C^, Б^b)
     | C^' ← 0
     | for b ∈ X/N + 1 .. ℵC
     |     | C^' ← Б^b  if C^' = 0
     |     | C^' ← stack(C^', Б^b) otherwise
     |     | if mod(b, X/N) = 0
     |     |     | C^ ← augment(C^, C^')
     |     |     | C^' ← 0
     | C^ ← (C^ + |min(C^)|) / max(C^ + |min(C^)|) · 255
```

(М.69)

Результаты встраивания ЦВЗ в контейнер путем модификации отношения между значениями коэффициентов соседних блоков и в пределах одного блока приведены, соответственно, на рис. 5.40 *а* и *б*.


**Рис. 5.40. Контейнеры с ЦВЗ, встроенным по формулам (5.29) (*а*) и (5.33) (*б*)**

**Шаг 15**

Рассмотрим процесс извлечения ЦВЗ из изображения-контейнера. Как было указано выше, для извлечения ЦВЗ помимо контейнера с, возможно, встроенным водяным знаком (**C***), необходимо наличие оригинального (незаполненного) контейнера (**C**) и изображения ЦВЗ (**W**).

Изображения **C** и **C*** разбиваем на блоки, используя программный модуль (М.55). Заметим, что при разбитии **C*** в модуле необходимо провести соответствующую замену переменных на такие, которые характеризуют именно это изображение.

К каждому блоку оригинального изображения и изображения, исследуемого на наличие ЦВЗ, применяем операцию прямого ДКП (модули (М.46,47)). На основе полученных матриц коэффициентов ДКП ($\Omega$ и $\Omega^*$) формируем матрицы СЧ-коэффициентов $\Omega_{mid}$ и $\Omega^*_{mid}$ (модуль (М.63)), которые используем при вычислении шаблонов полярности $P$ и $P^*$ (модуль (М.64)).

Путем поблочного сложения по модулю 2 полученных матриц полярностей получаем двоичные данные, которые, в том случае, если контейнер действительно содержит ЦВЗ, отвечают переставленным в пространстве и псевдослучайно смешанным элементам ЦВЗ. Операцию суммирования выполняем с помощью программного модуля (М.70).

$$\begin{aligned} &W^*_{sort} \leftarrow \text{for } b \in 1 .. \aleph_{C^*} \\ &\quad W^*sort_b \leftarrow \overrightarrow{\left(P_b \oplus P^*_b\right)} \\ &W^*_{sort} \end{aligned} \qquad \text{(М.70)}$$

Формируем массив $T_\Sigma$ индексов сопоставленных пар блоков контейнера и оригинального ЦВЗ (см. (М.59), (М.60)), на основании которого выполняем обратную пространственную перестановку блоков массива $W^*_{sort}$ — программный модуль (М.71).

$$\begin{aligned} &Б^*W \leftarrow \text{for } b \in 1 .. \aleph_{C^*} \\ &\quad s1 \leftarrow T_{\Sigma_{b,1}} \\ &\quad s2 \leftarrow T_{\Sigma_{b,2}} \\ &\quad Б^*W_{s2} \leftarrow W^*sort_{s1} \\ &Б^*W \end{aligned} \qquad \text{(М.71)}$$

Пространственно переставленные блоки $Б^*_W$ объединяем в общий массив $W^*_{rnd}$ (М.72), элементы которого гипотетически являются псевдослучайно перемешанными элементами оригинального ЦВЗ.

$$\begin{aligned} &W^*_{rnd} \leftarrow W^*_{rnd} \leftarrow Б^*W_1 \\ &\text{for } b \in 2 .. A \div n \\ &\quad W^*_{rnd} \leftarrow \text{stack}\left(W^*_{rnd}, Б^*W_b\right) \\ &W^{*\prime}_{rnd} \leftarrow 0 \\ &\text{for } b \in A \div n + 1 .. \aleph_W \\ &\quad W^{*\prime}_{rnd} \leftarrow Б^*W_b \text{ if } W^{*\prime}_{rnd} = 0 \\ &\quad W^{*\prime}_{rnd} \leftarrow \text{stack}\left(W^{*\prime}_{rnd}, Б^*W_b\right) \text{ otherwise} \\ &\quad \text{if } \text{mod}(b, A \div n) = 0 \\ &\quad\quad W^*_{rnd} \leftarrow \text{augment}\left(W^*_{rnd}, W^{*\prime}_{rnd}\right) \\ &\quad\quad W^{*\prime}_{rnd} \leftarrow 0 \\ &W^*_{rnd} \end{aligned} \qquad \text{(М.72)}$$

Используя программный модуль (М.73), проводим обратную псевдослучайную перестановку данных, используя (М.57) для получения ПСЧ, на основе которых генерируется пара координат элемента в массиве $W^*_{rnd}$, значение которого присваивается $i$-му элементу вектора $W^*_{vec}$.

$$W^*_{vec} := \left| \begin{array}{l} \Im \leftarrow \mathbf{Vrnd}(s) \\ \mathbf{for}\ \ i \in 1..A \cdot Z \\ \quad \left| \begin{array}{l} a \leftarrow \mathbf{trunc}\left(\dfrac{\Im_i - 1}{A}\right) + 1 \\ z \leftarrow \mathbf{mod}(\Im_i, Z) + 1 \\ W^*\mathrm{vec}_i \leftarrow W^*\mathrm{rnd}_{a,z} \end{array} \right. \\ W^*_{vec} \end{array} \right. \tag{M.73}$$

Полученный в результате выполнения (М.73) вектор сворачиваем в массив **W***, имеющий размерность оригинального ЦВЗ (модуль (М.74)).

$$W^* := \mathbf{for}\ \ z \in 1..Z \quad W^{*\langle z \rangle} \leftarrow \mathbf{submatrix}\left[W^*_{vec}, (z-1)\cdot A + 1, z\cdot A, 1, 1\right] \tag{M.74}$$

Графическое представление извлеченных ЦВЗ, встраивание которых в контейнер было проведено путем изменения отношений между значениями коэффициентов ДКП, соответственно, соседних блоков и в пределах одного блока, изображено на рис. 5.41 *а* и *б*.

Результаты вычисления показателей визуального искажения для двух рассмотренных разновидностей метода занесены в табл.5.4. Заметим, что при сравнении полученных результатов с результатами других методов, следует принимать во внимание, что в контейнер встраивалась информация, пиксельный объем которой был всего лишь в 4 раза меньше объема контейнера.


а) б)

**Рис. 5.41. ЦВЗ, извлеченные из контейнера C^**

#### 5.3.3.4. Метод Фридрих

Алгоритм, предложенный Джессикой Фридрих (J. Fridrich) [106], по сути является комбинацией двух алгоритмов: в соответствии с одним из них скрываемые данные встраиваются в низкочастотные, а с другим — в среднечастотные коэффициенты ДКП. Как было показано автором, каскадное использование двух разных алгоритмов позволяет получить хорошие результаты относительно стойкости стеганографической системы к атакам.

Изображение, которое планируется использовать в качестве контейнера, конвертируется в сигнал с нулевым математическим ожиданием и определенным стандартным отклонением таким образом, чтобы НЧ-коэффициенты ДКП, которые будут вычислены в дальнейшем, попадали в предварительно заданный неизменный диапазон. Предложенное преобразование

$$\boldsymbol{G} = \frac{1024}{\sqrt{\boldsymbol{X}\cdot\boldsymbol{Y}}} \cdot \frac{\boldsymbol{C} - \overline{\boldsymbol{C}}}{\boldsymbol{\sigma}(\boldsymbol{C})}, \tag{5.37}$$

где $\boldsymbol{X}$, $\boldsymbol{Y}$ — размерность изображения $\boldsymbol{C}$ в пикселях; $\overline{\boldsymbol{C}}$ и $\boldsymbol{\sigma}(\boldsymbol{C})$ — соответственно, математическое ожидание и стандартное отклонение значений яркости пикселей изображения, — трансформирует полутоновое изображение $\boldsymbol{C}$ в двумерный сигнал $\boldsymbol{G}$ с нулевым математическим ожиданием. В этом случае абсолютное значение максимального НЧ-коэффициента ДКП сигнала $\boldsymbol{G}$ не будет превышать порог

(200…250). Кроме того, утверждается, что данное преобразование применимо для широкого круга всевозможных изображений: как с большими однородными областями, так и сильно текстурированных.

Для сигнала-изображения $\boldsymbol{G}$ проводится вычисление коэффициентов ДКП, из всего множества которых модифицируются только низкочастотные. Причем модифицирование проводится таким образом, чтобы в коэффициентах было закодировано скрываемое сообщение $\boldsymbol{W}$, представляющее собой сигнал в виде последовательности чисел $\{-1, 1\}$. Для этого предварительно необходимо определить геометрическую прогрессию действительных чисел

$$\tau_{i+1} = \frac{1+\alpha}{1-\alpha} \cdot \tau_i; \quad \tau_1 = 1, \tag{5.38}$$

параметризованную (настраиваемую) с помощью параметра $\alpha \in (0, 1)$.

Для значений $t > 1$, $\tau_i \le t < \tau_{i+1}$ определяется индексная функция

$$ind(t) = (-1)^i, \text{ если } t \in [\tau_i, \tau_{i+1}), \tag{5.39}$$

позволяющая для каждого действительного числа $t > 1$ определить его индекс ($+/-1$). Вполне очевидно, что указанный индекс может быть изменен путем прибавления или же вычитания числа, не превышающего значения $\alpha \cdot t$. На рис. 5.42 приведены индексные функции для $\alpha$ = 0.1, 0.2 и 0.3.

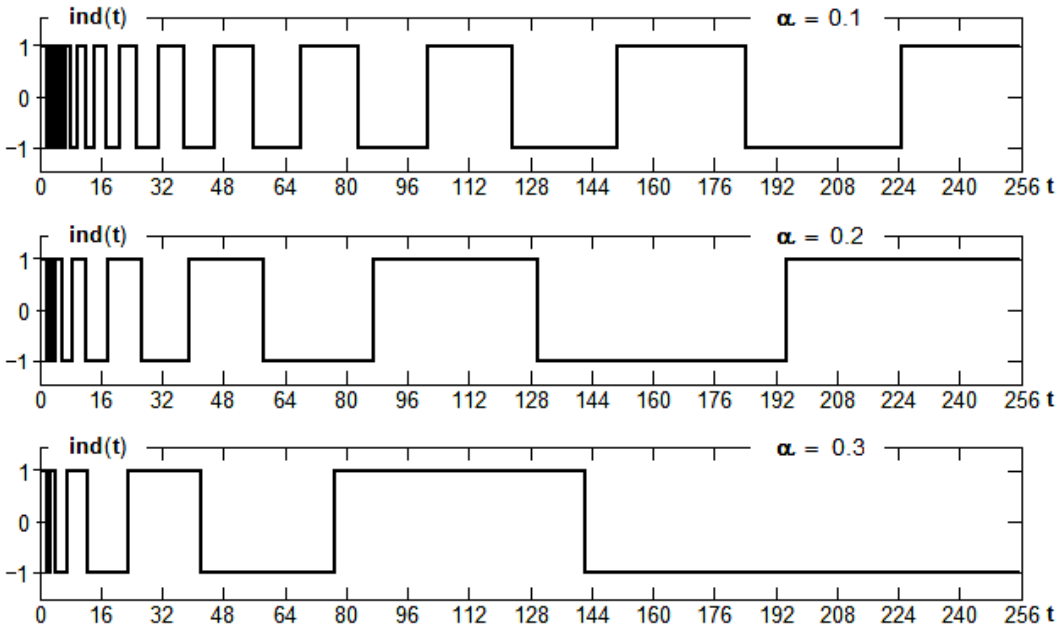



**Рис.5.42. Индексная функция ind(t) при значениях $\alpha$ = 0.1, 0.2 и 0.3**

Для встраивания массива сообщения $\boldsymbol{W}$, каждый отдельный бит которого может принимать значения $W_j \in \{-1, 1\}$, $j \in \{1, 2, ..., \aleph_W\}$, выбираются $\aleph_{\Omega low} = \aleph_W$ НЧ-коэффициентов ДКП — $\Omega_j$, значения которых изменяются таким образом, чтобы удовлетворялось условие $ind\left(\left|\Omega\hat{}_j\right|\right) = W_j$, где $\Omega\hat{}_j$ — модифицированное значение коэффициента ДКП. В том случае, если $|\Omega_j| < 1$, коэффициент для встраивания не используется.

Благодаря свойствам индексной функции, как указывает автор, каждый коэффициент будет изменен не больше чем на $100\cdot\alpha$ процентов[9]. Также отмечается, что изменения будут носить случайных характер, поскольку не существует никаких оснований считать, что коэффициенты ДКП на начальном этапе кодирования являются следствием определенного сообщения.

Наибольшая стойкость стеганосистемы к искажениям контейнера достигается при установлении в качестве новых значений коэффициентов ДКП середины интервалов $[\tau_i, \tau_{i+1})$. Однако это может послужить появлению сосредоточений одинаковых коэффициентов ДКП, что делает систему ненадежной с точки зрения возможного стеганографического анализа. Значение параметра $\alpha$ выбирается таким образом, чтобы встраивание сообщения не приводило к заметным для глаза искажениям контейнера.

Операция извлечения проводится путем выполнения аналогичных с операцией встраивания преобразований контейнера, который подозревается на наличие скрытого сообщения:

- конвертация в сигнал с нулевым матожиданием по формуле (5.37);
- вычисление коэффициентов ДКП конвертированного изображения;
- вычисление для заранее оговоренных коэффициентов ДКП индексной функции (5.39) при заданном параметре $\alpha$;
- формирование из полученных индексов массива извлеченного сообщения.

---

[9] Более точно максимально возможное изменение, которое может претерпеть коэффициент ДКП, можно вычислить, рассмотрев следующий предельный случай. Пусть первичное значение коэффициента ДКП попадает в интервал $[\tau_b, \tau_{b+1})$, как можно ближе к $\tau_{b+1}$ (см. рис. 5.43, значения индексной функции условны). Также предположим, что в результате случайного выбора из интервала $[\tau_{b-1}, \tau_b)$ (считаем, что изменение происходит в сторону меньших значений) первичному значению коэффициента было присвоено значение $\tau_{b-1}$.

При этом близкое к $\tau_{b+1}$ значение ($\tilde{\tau}_{b+1}$) превышает $\tau_{b-1}$ в $\left(\frac{1+\alpha}{1-\alpha}\right)^2$ раз или же на $\left[\left(\frac{1+\alpha}{1-\alpha}\right)^2 - 1\right]\cdot 100 \Big/ \left(\frac{1+\alpha}{1-\alpha}\right)^2$ процентов. График зависимости максимально возможного изменения коэффициента ДКП от параметра $\alpha$ приведен на рис. 5.44.

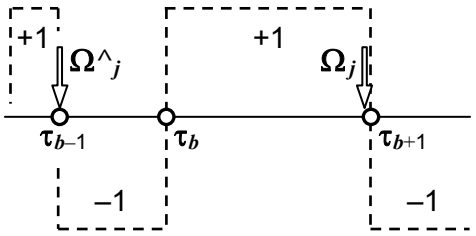



**Рис. 5.43. К объяснению определения максимально возможного изменения, которое может претерпеть коэффициент ДКП**

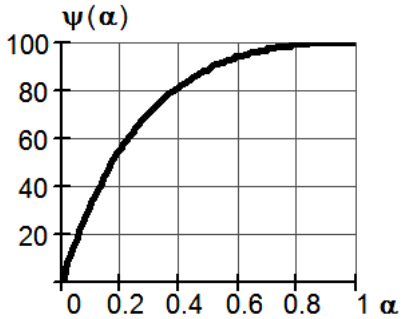



**Рис.5.44. Зависимость максимально возможного изменения коэффициента ДКП от параметра α**

Кроме того, Фридрих предложен метод детектирования наличия/отсутствия встроенного сообщения в контейнере, что может быть полезным при защите цифрового контента (информационного содержимого) с помощью ЦВЗ. Данная операция предполагает осведомленность получателя относительно содержания скрытого сообщения.

По причине того, что большинство из $\aleph_{\Omega low}$ НЧ-коэффициентов было подвергнуто модификации во время кодирования, простое вычисление корреляции между $W_j$ и $ind\left(\left|\Omega\hat{}_j\right|\right)$ предопределяло бы собой нестойкость метода, поскольку малые, визуально незначащие коэффициенты ДКП делают такой же весовой вклад в общую энергию сигнала, как и большие, визуально более значащие коэффициенты.

Так как предварительно было выдвинуто условие, что контейнер со встроенным сообщением не должен привлекать внимание, мы не можем встраивать данные только в коэффициенты, имеющие большое значение. Кроме того, позиции наибольших коэффициентов ДКП первичного и модифицированного изображений могут не совпадать, что сделает невозможной безошибочную идентификацию тех из них, в которые было произведено встраивание. В предлагаемой автором системе встраивание осуществляется во все НЧ-коэффициенты, независимо от их значения (обычно кроме тех, что не превышают единицы), однако лишь наибольшие из них учитываются в последствии при вычислении коэффициента корреляции, взвешиваемого с энергией абсолютных значений коэффициентов ДКП:

$$K = \frac{\sum_{j=1}^{\aleph_{\Omega\, low}} \left|\Omega_j^*\right|^{\beta} \cdot ind\left(\left|\Omega_j^*\right|\right) \cdot W_j}{\sum_{j=1}^{\aleph_{\Omega\, low}} \left|\Omega_j^*\right|^{\beta}}. \qquad (5.40)$$

Такое взвешивание автоматически делает более выразительными наибольшие значения коэффициентов, одновременно подавляя незначительные, которые могли претерпеть изменения в результате каких-либо операций по обработке изображения. Параметр $\beta$ устанавливает важность взвешивания. Если $\beta = 0$, вычисляется обычный, невзвешенный коэффициент корреляции. Значение $\beta$, слишком близкое к единице, приводит к сингулярности (вырождению) системы детектирования: функция обнаружения будет зависеть только от значения всего лишь одного бита, отвечающего наибольшему коэффициенту ДКП. Автор метода рекомендует использовать значения $\beta \in (0.5, 1)$.

Более стойкой к атакам данную систему можно сделать путем поиска максимального значения коэффициента корреляции относительно стандартного отклонения значений яркости пикселей изображения, подозреваемого на присутствие встроенного сообщения.

Масштабирование (5.37) зависит от стандартного отклонения значений яркости пикселей, которое может быть существенно искажено, если изображение с встроенным сообщением было подвергнуто сглаживанию или, например, дополнительно зашумлено. Как следствие, коэффициенты ДКП такого изображения будут промасштабированы с помощью фиксированного коэффициента (отношение стандартных

отклонений оригинального и исследуемого на наличие скрытого сообщения изображений, $d = \sigma(C)/\sigma(S)$). Однако сообщение, закодированное в коэффициентах ДКП, линейные изменения не коснутся. Указанный факт и наводит на мысль о целесообразности использования простого одномерного поиска правильного масштаба $d$, который бы максимизировал значение коэффициента корреляции (поскольку, как было отмечено выше, первичное изображение, используемое в качестве контейнера, в детекторе отсутствует). Таким образом, дополненная функция детектирования имеет следующий вид:

$$K' = \max_{d \in (1-\delta,\, 1+\delta)} K(d) = \frac{\sum_{j=1}^{\aleph_{\Omega\, low}} \left|\Omega_j^*\right|^{\beta} \cdot ind\left(d \cdot \left|\Omega_j^*\right|\right) \cdot W_j}{\sum_{j=1}^{\aleph_{\Omega\, low}} \left|\Omega_j^*\right|^{\beta}}. \qquad (5.41)$$

Автором [106] установлено, что даже при значительных искажениях изображения в результате атак, достаточным будет являться шаг отклонения масштаба $\delta = 0{,}25$.

Трудности детектирования, возникающие при этом, требуют уменьшения информационного содержания сообщения, и добавления корректирующих бит. Следовательно, поскольку вклад в обнаружение сообщения вносят только наибольшие коэффициенты ДКП, информационное содержание сообщения длиной $\aleph_W$ представляет собой лишь определенную долю от $\aleph_W$. Кроме того, вполне очевидно, что одномерный поиск масштабного коэффициента, который максимизирует коэффициент корреляции, увеличивает процент ошибочных обнаружений.

Для достижения свойств высокой устойчивости к атакам на стеганосистему при одновременном наименьшем (насколько, конечно, это возможно) искажении контейнера Фридрих было предложено встроить в последний дополнительное сообщение, используя методику расширения спектра[10]. При этом встраивание сообщения осуществляется путем добавления шумоподобного сигнала к СЧ-коэффициентам ДКП изображения. Количество таких коэффициентов ($\aleph_{\Omega mid}$) составляет приблизительно 30% от общего количества коэффициентов ДКП.

Считается, что информация, которую несет дополнительное сообщение, состоит из $\aleph_{W^+}$ символов $W_j^+$, каждый из которых может быть представлен десятичным целым числом, $1 \le W_j^+ \le \max(W^+)$.

Для каждого $j$-го символа генерируется последовательность $\xi^{(j)}$ ПСЧ, равномерно распределенных в интервале [0, 1]. Начальное состояние генератора ПСЧ может выступать в роли секретного ключа. Мощность $j$-го множества ПСЧ: $\left|\xi^{(j)}\right| \ge \aleph_{\Omega\, mid} + \max(W^+)$.

Для представления отдельного символа сообщения $W^+$, из множества $\xi^{(j)}$ выделяется сегмент $\eta^{(j)} = \xi^{(j)}_{W_j^+}, \ldots, \xi^{(j)}_{W_j^+ + \aleph_{\Omega\, mid} - 1}$, который содержит $\aleph_{\Omega mid}$ элементов.

[10] Например, методом прямой последовательности.

В результате, сообщение из $\aleph_{W^+}$ символов может быть представлено в виде следующей суммы[11]:

$$Spr = \frac{\left[\sum_{j=1}^{\aleph_{W^+}} \eta^{(j)}\right] - \frac{\aleph_{W^+}}{2}}{\sqrt{\aleph_{W^+}/12}}. \qquad (5.42)$$

Сигнал с расширенным спектром ***Spr*** характеризуется приблизительно нормальным (гауссовым) распределением с нулевым математическим ожиданием и единичным стандартным отклонением (точность аппроксимации возрастает при увеличении значения $\aleph_{W^+}$). В дальнейшем сигнал ***Spr*** умножается на параметр $\gamma$, который регулирует отношение «устойчивость/заметность встраивания» и поэлементно суммируется с $\aleph_{\Omega mid}$ выбранными СЧ-коэффициентам.

Извлечение сообщения $W^+$ выполняется путем предварительного вычисления коэффициентов ДКП изображения и выделения среди них именно среднечастотных (данная операция должна быть согласованной с соответствующим действием на этапе встраивания). Используя секретный ключ/алгоритм, осуществляется генерация последовательностей ПСЧ (общим количеством $\aleph_{W^+}$, если данный параметр известен; в противном случае — по обстоятельствам, исходя из анализа уже извлеченной части сообщения) длиной $\aleph_{\Omega\, mid} + \max(W^+)$.

Из каждой последовательности $\xi^{(j)}$ выделяется $\max(W^+)$ сегментов длиной $\aleph_{\Omega mid}$ элементов, для которых рассчитывается взаимная корреляция с вектором выделенных СЧ-коэффициентов. Позиция наибольшего значения функции корреляции в полученном при этом векторе и будет определять значение, которое предположительно имел встроенный символ $W_j^+$.

Рассмотрим реализацию данного метода на практике.

**Шаг 1**

Пусть изображение-контейнер и изображение-ЦВЗ представлены графическими файлами C.bmp и W.bmp соответственно (рис. 5.45):

**C** := **READBMP**("C.bmp");
**W** := **READBMP**("W.bmp").

При этом характеристики, отвечающие указанным изображениям, составляют:

**X** := **rows**(**C**), **X** = 256; **Y** := **cols**(**C**), **Y** = 256;
**A** := **rows**(**W**), **A** = 16; **Z** := **cols**(**W**), **Z** = 16.

[11] Следует заметить, что для формирования нормального распределения на основе равномерного в [106] предлагается использовать выражение $Spr = \sum_{j=1}^{\aleph_{W^+}} \eta^{(j)} \Big/ \sqrt{\aleph_{W^+}}$, которое, как известно из теории статистических распределений, не отвечает поставленным требованиям (см., например, [107]).

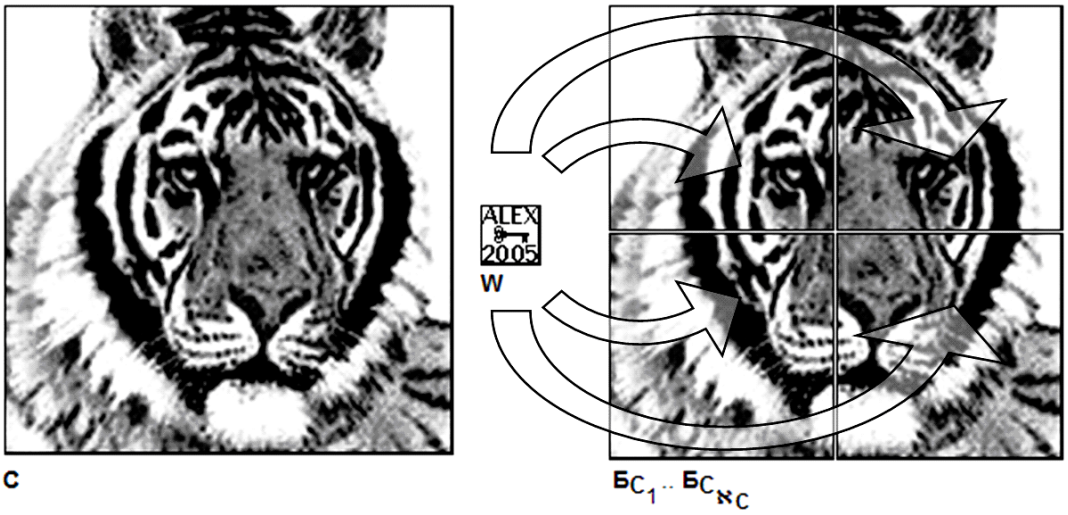



**Рис. 5.45. Пример контейнера-оригинала (C), ЦВЗ (W) и контейнера, разбитого на $\aleph_C$ блоков $Б_C$**

**Шаг 2**

Для большей стойкости стеганосистемы, контейнер разбиваем на блоки размерностью **N**×**N**, при **N** := 128 (см. программный модуль (М.55)). Количество блоков — $\aleph_C := X \cdot Y \div N^2$, $\aleph_C = 4$.

**Шаг 3**

Используя первый (низкочастотный) алгоритм Фридрих, в каждый из блоков будем встраивать один и тот же ЦВЗ.

Трансформацию (5.37) реализуем отдельно для каждого блока с помощью программного модуля (М.75). Функции **mean**($Б_{C_b}$) и **stdev**($Б_{C_b}$) возвращают, соответственно, среднее значение и стандартное отклонение для элементов массива $Б_{C_b}$.

$$Б_G := \left| \begin{array}{l} \textbf{for}\ \ b \in 1 .. \aleph_C \\ \quad Б_{G_b} \leftarrow \dfrac{1024}{N} \cdot \dfrac{Б_{C_b} - \textbf{mean}\left(Б_{C_b}\right)}{\textbf{stdev}\left(Б_{C_b}\right)} \\ Б_G \end{array} \right. \qquad \text{(М.75)}$$

Формирование прогрессии (5.38) осуществляется программным модулем (М.76).

$$\tau := \left| \begin{array}{l} \textbf{for}\ \ i \in 1 .. 300 \\ \quad \left| \begin{array}{l} \tau_i \leftarrow \left(\dfrac{1+\alpha}{1-\alpha}\right)^{i-1} \\ \textbf{break}\ \ \textbf{if}\ \ \tau_i > 256 \end{array} \right. \\ \tau \end{array} \right. \qquad \text{(М.76)}$$

Верхняя граница переменной цикла **i** (300) и прерывание цикла в случае превышения элементом $\tau_i$ порога 256 выбрано условно, исходя из того, что в результате трансформации (М.75) наибольшие значения НЧ-коэффициентов ДКП не будут превышать 200...250. Кроме того, для прохождения всех 300 отсчетов значение па-

раметра $\alpha$ должно быть меньше 0.01, что на практике не используется из-за низкой стойкости к атакам полученных при указанных условиях стеганосистем.

Вид прогрессии $\tau$ при $\alpha = 0.1$ приведен на рис.5.46.

$\tau =$

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 6 | 2.727 | 11 | 7.439 | 16 | 20.289 | 21 | 55.335 | 26 | 150.923 |
| 2 | 1.222 | 7 | 3.334 | 12 | 9.092 | 17 | 24.797 | 22 | 67.632 | 27 | 184.461 |
| 3 | 1.494 | 8 | 4.074 | 13 | 11.112 | 18 | 30.308 | 23 | 82.662 | 28 | 225.452 |
| 4 | 1.826 | 9 | 4.980 | 14 | 13.582 | 19 | 37.043 | 24 | 101.031 | 29 | 275.553 |
| 5 | 2.232 | 10 | 6.086 | 15 | 16.600 | 20 | 45.274 | 25 | 123.482 | 30 | |

**Рис. 5.46. Прогрессия $\tau$ при параметре $\alpha = 0.1$**

Программный модуль (М.77) определяет индексную функцию для аргумента **t**. В предложенном варианте последняя определена и для значений аргумента $0 \le t < 1$. Примеры индексных функций для некоторых значений параметра $\alpha$ приведены на рис.5.42.

$$\begin{aligned} &\mathbf{ind}(t) := \\ &\quad \text{for } i \in 1..\mathbf{rows}(\tau) - 1 \\ &\quad\quad \mathbf{ind} \leftarrow 1 \quad \text{if } 0 \le t < \tau_1 \\ &\quad\quad \mathbf{ind} \leftarrow (-1)^i \quad \text{if } \tau_i \le t < \tau_{i+1} \\ &\quad \mathbf{ind} \end{aligned} \qquad \text{(М.77)}$$

**Шаг 4**

На данном этапе необходимо провести ДКП трансформированных блоков изображения. Очевидно, что время, необходимое для вычисления матрицы ДКП одного блока изображения по формулам (5.18), существенно зависит от размерности блока. Для вычисления полной матрицы ДКП используются два вложенных цикла перебирания индексов элементов матрицы, общее количество которых равняется $N^2$ (см., например, (М.47) и (М.49)).

Для вычисления одного элемента матрицы необходимо выполнить вычисление двух вложенных суммирований ($N^2$ операций), внутри которых вычислить аргументы двух косинусов ($2 \cdot 6$ операций), собственно косинусы ($2 \cdot 1$ операции) и два произведения. Кроме того, полученный при этом результат (обозначим его как $\Sigma\Sigma$) умножается на коэффициент: $\frac{\zeta(u) \cdot \zeta(v)}{\sqrt{2 \cdot N}} \cdot \Sigma\Sigma$, — еще 5 операций. Следовательно, общее количество шагов или арифметико-логических операций, необходимых для решения данной вычислительной проблемы (формирование матрицы ДКП одного блока) составляет:

$$\mathbf{O}(N) = N^2 \cdot \left[ 5 + N^2 \cdot (2 \cdot 6 + 2 \cdot 1 + 2) \right] = N^2 \cdot \left( 5 + 16 \cdot N^2 \right). \qquad (5.43)$$

Соответствующим образом возрастает и время вычисления алгоритма. На рис. 5.47 приведен график зависимости времени вычисления программного модуля (М.47) ДКП блока изображения размерностью **N**×**N** от значения **N**, полученный с помощью вычислительной системы следующей конфигурации: процессор Intel Pentium 4HT 2.4 ГГц, FSB 4×200 МГц, Dual DDR SDRAM 2×256 Мб.

Таким образом, в нашем случае, при проведении прямого ДКП по модулю (М.47), реализующего формулу (5.18*a*), время вычисления всех $\aleph_C = 4$ блоков размерностью 128×128 составило бы около 40 минут (по 10 минут на блок).

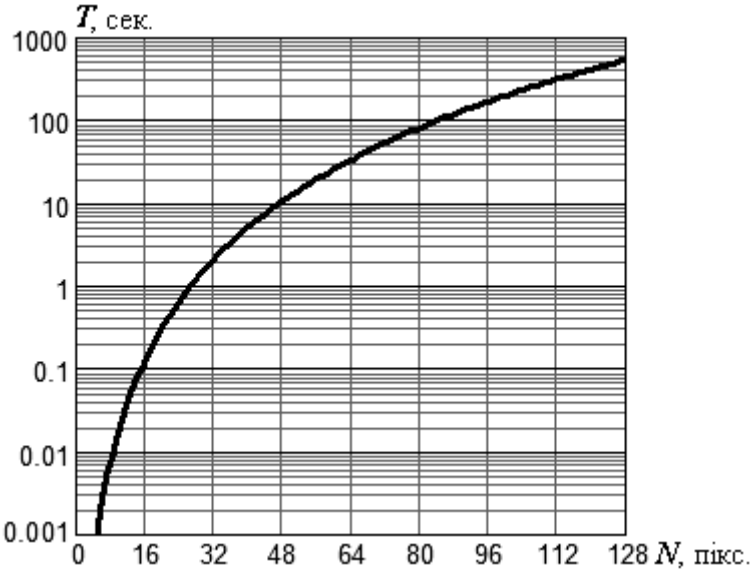



**Рис. 5.47. Зависимость продолжительности *T* проведения ДКП блока от его размерности *N* при использовании для вычисления программного модуля (М.47)**

Значительно более эффективный вариант вычисления коэффициентов ДКП реализуется через произведение матриц. При таком подходе формула прямого ДКП может быть представлена в следующем виде

$$\boldsymbol{\Omega} = \zeta \cdot \boldsymbol{C} \cdot \zeta^{\mathrm{T}}; \tag{5.44}$$

$$\zeta_{\upsilon,\nu} = \begin{cases} \dfrac{1}{\sqrt{N}}, \text{ при } \upsilon = 0,\ 0 \le \nu \le N-1; \\ \sqrt{\dfrac{2}{N}} \cdot \cos\left[\dfrac{\pi \cdot \upsilon \cdot (2 \cdot \nu + 1)}{2 \cdot N}\right], \text{ при } 1 \le \upsilon \le N-1,\ 0 \le \nu \le N-1, \end{cases} \tag{5.45}$$

где $\boldsymbol{\Omega}$ — матрица коэффициентов ДКП; $\zeta$ — трансформационная матрица ДКП размерностью $N \times N$, элементы которой определяются по формуле (5.45); $\boldsymbol{C}$ — матрица яркостей пикселей изображения размерностью $N \times N$; $\zeta^{\mathrm{T}}$ – транспонированная матрица $\zeta$. В (5.44) результатом произведения матриц $\zeta \cdot \boldsymbol{C}$ является матрица размерностью $N \times N$, чьи столбцы содержат результат одномерного ДКП столбцов $\boldsymbol{C}$. Произведение полученной матрицы на $\zeta^{\mathrm{T}}$ позволяет получить двумерное ДКП. Поскольку $\zeta$ представляет собой ортонормированную матрицу действительных элементов, транспонированная матрица является эквивалентной обратной: $\zeta^{\mathrm{T}} = \zeta^{-1}$. Таким образом, формула обратного ДКП:

$$\boldsymbol{S} = \zeta^{-1} \cdot \boldsymbol{\Omega} \cdot \zeta = \zeta^{\mathrm{T}} \cdot \boldsymbol{\Omega} \cdot \zeta. \tag{5.46}$$

При произведении двух матриц «цена» вычисления одного элемента результирующей матрицы составляет $N$ произведений и $N$ суммирований, а при вычислении полной матрицы — $(N+N) \cdot N^2 = 2 \cdot N^3$. Поскольку в (5.44), (5.46) присутствует по два произведения, вычисление матрицы $\boldsymbol{\Omega}$ будет состоять из $4 \cdot N^3$ шагов:

$$\mathrm{O}'(N) = 4 \cdot N^3. \tag{5.47}$$

По сравнению с (5.43) это существенное повышение скорости вычисления. Так, например, в случае $N = 8$ время вычисления сокращается приблизительно в 32 раза, при $N = 128$ выигрыш составляет уже около 512.

Программный модуль прямого ДКП трансформированных блоков изображения — (М.78).

$$\zeta := \left| \begin{array}{l} \textbf{for}\ \ \upsilon \in 0..\mathbf{N}-1 \\ \quad \textbf{for}\ \ \nu \in 0..\mathbf{N}-1 \\ \quad\quad \left| \begin{array}{l} \zeta_{\upsilon+1,\nu+1} \leftarrow \frac{1}{\sqrt{\mathbf{N}}} \quad \textbf{if}\ \ \upsilon = 0 \\ \zeta_{\upsilon+1,\nu+1} \leftarrow \sqrt{\frac{2}{\mathbf{N}}} \cdot \cos\left[\frac{\pi \cdot \upsilon \cdot (2 \cdot \nu + 1)}{2 \cdot \mathbf{N}}\right] \quad \textbf{if}\ \ 1 \le \upsilon \le \mathbf{N}-1 \end{array} \right. \\ \zeta \end{array} \right. \qquad \text{(М.78)}$$

$$\Omega := \left| \begin{array}{l} \textbf{for}\ \ \mathbf{b} \in 1..\aleph_C \\ \quad \Omega_{\mathbf{b}} \leftarrow \zeta \cdot Б G_{\mathbf{b}} \cdot \zeta^{T} \\ \Omega \end{array} \right.$$

График зависимости времени вычисления ДКП одного блока с помощью модуля (М.78) приведен на рис. 5.48.

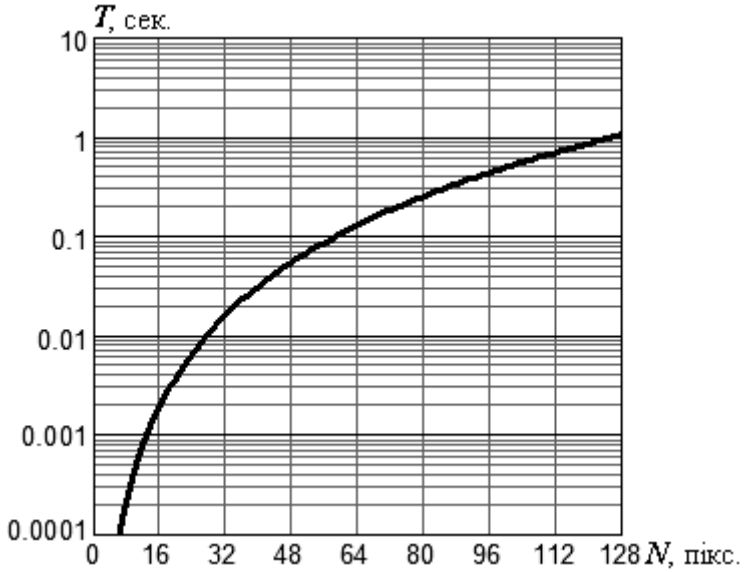



**Рис. 5.48. Зависимость продолжительности *T* проведения ДКП блока от его размерности *N* при использовании для вычисления программного модуля (М.78)**

**Шаг 5**

Для удобства встраивания, развернем массив ЦВЗ в вектор, используя программный модуль (М.56). Очевидно, что общее количество элементов полученного вектора $\mathbf{W_{vec}}$ будет равняться $\aleph_W := \mathbf{A} \cdot \mathbf{Z}$; $\aleph_W = 1024$.

**Шаг 6**

Выбор НЧ-коэффициентов ДКП из матрицы $\mathbf{\Omega_b}$ будем проводить исходя из того, что для элементов матрицы, которые находятся выше побочной диагонали (НЧ-

коэффициенты), сумма индексов меньше чем **N**+1, а для тех, которые ниже (ВЧ-коэффициенты) — больше чем **N**+1.

Введем метки **L** и **H**, которые будут определять диапазон суммы индексов. Элемент матрицы $\Omega_{\mathbf{b}}$, сумма индексов которого не выходит за установленные пределы, будет выбираться для встраивания.

Программный модуль поиска метки **H** при заданной метке **L** (М.79) построен на основе подсчета количества элементов, сумма индексов которых будет удовлетворять поставленному требованию невыхода за пределы диапазона (**L**, **H**).

$$
\begin{array}{l}
\mathbf{H} := \left| \begin{array}{l}
\mathbf{H} \leftarrow \mathbf{L} \\
\quad \mathbf{\Sigma bit} \leftarrow \sum_{\mathbf{i}=\mathbf{L}-1}^{\mathbf{H}-3} (1+\mathbf{i}) \\
\quad \mathbf{while}\ \ \mathbf{\Sigma bit} < \aleph_{\mathbf{W}} \\
\quad\quad \left| \begin{array}{l}
\mathbf{H} \leftarrow \mathbf{H}+1 \\
\quad \mathbf{\Sigma bit} \leftarrow \sum_{\mathbf{i}=\mathbf{L}-1}^{\mathbf{H}-3} (1+\mathbf{i})
\end{array} \right. \\
\mathbf{H}
\end{array} \right.
\end{array}
\qquad \text{(М.79)}
$$

Верхний предел **H** увеличивается до того момента, пока общее количество указанных элементов не превысит общего количества $\aleph_{\mathbf{W}}$ элементов вектора $\mathbf{W}_{\mathbf{vec}}$. Например, для **L** := 40 модуль (М.79) возвращает значение **H** = 62, что позволяет использовать для скрытия 1050 НЧ-коэффициентов каждой из матриц $\Omega_{\mathbf{b}}$.

Встраивание ЦВЗ в выбранные НЧ-коэффициенты блоков изображения реализует программный модуль (М.80). В том случае, если индексная функция **ind** от абсолютного значения **d** НЧ-коэффициента отвечает **j**-му элементу вектора ЦВЗ ($\mathbf{ind}(\mathbf{d}) = \mathbf{W}_{\mathbf{vec_j}}$), проводится поиск элементов прогрессии $\tau$, в интервале между которыми находится текущее значение **d**. Если данное значение является слишком близким к одной из границ интервала (а это, как указывалось выше, снижает стойкость создаваемой стеганосистемы), коэффициенту присваивается новое значение — первый член вектора элементов, распределенных, например, по нормальному закону при математическом ожидании $(\mathbf{t1} + \mathbf{t2})/2$ и стандартном отклонении $(\mathbf{t2} - \mathbf{t1})/13$. Отмеченное стандартное отклонение позволяет системе быть адаптированной к различным интервалам $(\tau_i, \tau_{i+1})$, а также предотвратить случай, когда и новое значение коэффициента снова окажется близким к границе интервала. В свою очередь, выбор случайного числа из определенного интервала (а не, например, значения, которое отвечает середине интервала) делает невозможным образование подозрительных сосредоточений одинаковых значений коэффициентов ДКП.

Если индексная функция $\mathbf{ind}(\mathbf{d}) \neq \mathbf{W}_{\mathbf{vec_j}}$, коэффициенту присваивается случайное значение из ближайшего интервала — $(\tau_i, \tau_{i+1})$, при $\mathbf{i} \le 2$; и $(\tau_{i-2}, \tau_{i-1})$, при $\mathbf{i} > 2$.

На рис. 5.49 схематически изображены массивы ДКП четырех блоков изображения. При этом, темные шумоподобные диагональные полосы в верхних левых углах блоков отвечают модифицированным НЧ-коэффициентам, а светлые элементы массивов — не модифицированным.

```
Ω^ := | for b ∈ 1 .. ℵC
      |   j ← 1
      |   Ω^' ← Ω_b
      |   for υ ∈ 1 .. N
      |     break if j > ℵW
      |     for ν ∈ 1 .. N
      |       if L < (υ + ν) < H
      |         d ← |Ω^'_{υ,ν}|
      |         # ← 1 if Ω^'_{υ,ν} ≥ 0
      |         # ← −1 if Ω^'_{υ,ν} < 0
      |         if ind(d) = W_vec_j
      |           for i ∈ 1 .. rows(τ)
      |             if d < τ_i
      |               if i = 1
      |                 t1 ← 0
      |                 t2 ← τ_i
      |               if i > 1
      |                 t1 ← τ_{i−1}
      |                 t2 ← τ_i
      |               break
      |           if (|d − t1| < 0.01) ∨ (|d − t2| < 0.01)
      |             Ω^'_{υ,ν} ← #·rnorm(1, (t1 + t2)/2, (t2 − t1)/13)_1
      |         if ind(d) ≠ W_vec_j
      |           for i ∈ 1 .. rows(τ)
      |             if d < τ_i
      |               if i ≤ 2
      |                 t1 ← τ_i
      |                 t2 ← τ_{i+1}
      |               if i > 2
      |                 t1 ← τ_{i−2}
      |                 t2 ← τ_{i−1}
      |               break
      |           Ω^'_{υ,ν} ← #·rnorm(1, (t1 + t2)/2, (t2 − t1)/13)_1
      |         j ← j + 1
      |       break if j > ℵW
      |   Ω^_b ← Ω^'
      | Ω^
```

(M.80)

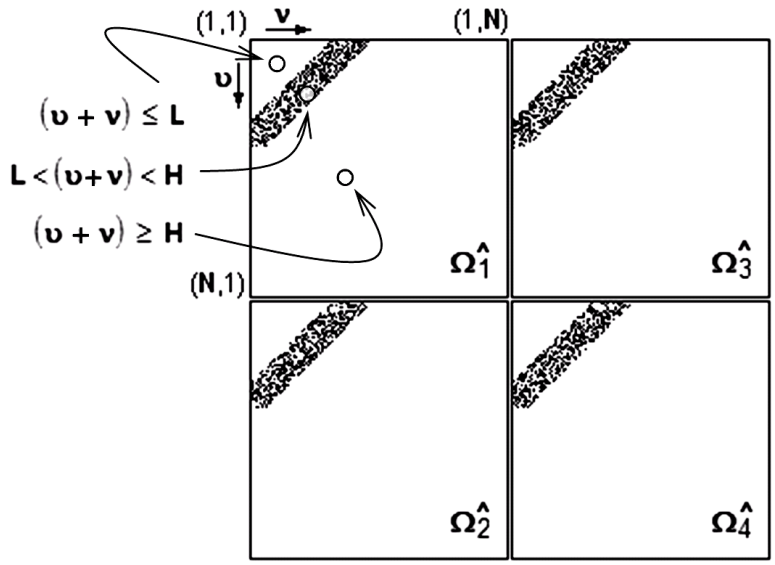



**Рис. 5.49. Массивы модифицированных (показано черным) НЧ-коэффициентов ДКП**

**Шаг 7**

Для реализации среднечастотного алгоритма Фридрих необходимо для каждого элемента ЦВЗ сгенерировать последовательность равномерно распределенных на интервале [0, 1] ПСЧ.

В качестве основы генератора ПСЧ можно использовать ЛРСОС, который реализован программным модулем (М.57), который генерирует действительные числа, равномерно распределенные в интервале $[1, 2^{d}]$. Для получения ПСЧ, равномерно распределенных в интервале [0, 1], достаточно разделить элементы полученного вектора **Vrnd**(•) на значение максимального из них ($2^{d}$), что и выполнено в программном модуле (М.81).

$$\begin{array}{l}\xi := \left|\begin{array}{l}\textbf{start} \leftarrow 74\\ \textbf{for}\ \ j \in 1..\aleph_W\\ \quad\left|\begin{array}{l}\xi' \leftarrow \textbf{Vrnd}(\textbf{start})\\ \textbf{start} \leftarrow \xi'_{74}\\ \xi_j \leftarrow \dfrac{\xi'}{\max(\xi')}\end{array}\right.\\ \xi\end{array}\right.\end{array} \qquad \text{(М.81)}$$

Модуль (М.81) позволяет сформировать для каждого из $\aleph_W^{+} = \aleph_W$ символов ЦВЗ ПСП длиной $2^{d}$ элементов. Начальное состояние (переменная **start**) генератора ПСЧ для 1-го символа ЦВЗ выбрано равным 74. В дальнейшем оно будет определяться значением элемента ПСП, который имеет индекс 74.

В процессе обработки метода установлено, что алгоритм расширения спектра наиболее надежен, если отношение $\left|\xi_j\right| / \aleph_W \geq 4$. Поэтому предварительно принимаем $\mathbf{d} := \mathbf{log}(4 \cdot \aleph_W, 2)$, $\mathbf{d} = 12$. При этом $2^{d} = 4096$.

Количество СЧ-коэффициентов ДКП блоков изображения, в которые будем встраивать ЦВЗ, примем равным $\aleph_{\Omega mid} := 4080$.

Элемент ЦВЗ, который равняется «+1», будем представлять случайным целым числом из диапазона [1, 7], а элемент, имеющий значение «–1» — случайным целым из диапазона [10, 16].

С учетом заданных выше начальных данных, проведем выделение сегмента $\eta_j$, который будет представлять **j**-й символ ЦВЗ, для чего воспользуемся программным модулем (М.82), в котором парой функций **round**(**runif**(•)) осуществляется генерирование случайного по равномерному закону целого числа в одном из двух диапазонов. С помощью функции **submatrix**(•) выполняется выделение из вектора $\xi_j$ сегмента длиной $\aleph_{\Omega mid}$.

```
η := | for j ∈ 1 .. ℵW+
     |   if Wvec_j = 1
     |     m ← round(runif(1, 1, 7)_1)
     |     ηj ← submatrix(ξj, m, ℵΩmid + m − 1, 1, 1)
     |   if Wvec_j = −1
     |     m ← round(runif(1, 10, 16)_1)
     |     ηj ← submatrix(ξj, m, ℵΩmid + m − 1, 1, 1)
     | η
```

(М.82)

Разделим ЦВЗ на $\aleph_C$ частей, каждую из которых представим в виде отдельной суммы в (5.42). Данная процедура реализована программным модулем (М.83).

$$Spr := \left| \begin{array}{l} \text{for } b \in 1 .. \aleph_C \\ \quad Spr_b \leftarrow \dfrac{\displaystyle\sum_{i=\frac{\aleph_W^+}{\aleph_C}\cdot(b-1)+1}^{\frac{\aleph_W^+}{\aleph_C}\cdot b} \eta_i \; - \; \frac{\aleph_W^+}{\aleph_C}\cdot\frac{1}{2}}{\sqrt{\frac{\aleph_W^+}{\aleph_C}\cdot\frac{1}{12}}} \\ Spr \end{array} \right. \qquad \text{(М.83)}$$

Результатом выполнения (М.83) являются векторы $Spr_b$, элементы которых должны иметь гауссово распределение с нулевым математическим ожиданием и единичным стандартным отклонением. В нашем случае для каждого из блоков были получены следующие характеристики:

| mean(Spr_b)= |
|---|
| 0.013 |
| 0.002 |
| -0.001 |
| -0.012 |

| stdev(Spr_b)= |
|---|
| 1.015 |
| 0.973 |
| 0.976 |
| 0.991 |

**Шаг 8**

Выбор из матрицы $\Omega_b$ СЧ-коэффициентов будем проводить, исходя из того, что для элементов, которые формируют побочную диагональ матрицы, сумма интервалов равняется $N+1$, сама же диагональ состоит из $N$ элементов.

Введем метки $L'$ и $H'$, которые будут определять диапазон суммы индексов, попадание в который будет относить рассматриваемый СЧ-коэффициент к разряду тех, в которые будет проводиться встраивание векторов $Spr_b$. При $\aleph_{\Omega mid} = 4080$ и $N = 128$ в результате использования программного модуля (М.84) получены следующие результаты: $L' = 111$, $H' = 147$, что делает доступными для встраивания 4174 СЧ-коэффициентов каждой из матриц $\Omega_b$.

```
(L')
(H') := | L' ← N
        | H' ← N + 2
        | Σbit ← N
        | j ← 1
        | while  Σbit < ℵΩmid
        |     | Σbit ← Σbit + 2·(N − j)
        |     | j ← j + 1
        |     | L' ← L' − 1
        |     | H' ← H' + 1
        | (L')
        | (H')
```

(М.84)

Встраивание элементов векторов $Spr_b$ в выбранные СЧ-коэффициенты блоков контейнера выполняет программный модуль (М.85).

```
Ω^^ := | for  b ∈ 1 .. ℵC
       |     | j ← 1
       |     | Ω^^' ← Ω^b
       |     | for  υ ∈ 1 .. N
       |     |     | break  if  j > ℵΩmid
       |     |     | for  ν ∈ 1 .. N
       |     |     |     | if  L' < (υ + ν) < H'
       |     |     |     |     | Ω^^'υ,ν ← Ω^^'υ,ν +
       |     |     |     |     |           + γ·(Spr_b)_j
       |     |     |     |     | j ← j + 1
       |     |     |     | break  if  j > ℵΩmid
       |     | Ω^^_b ← Ω^^'
       | Ω^^
```

(М.85)

Схематическое изображение результата встраивания (как отличия от оригинальных матриц $\Omega_b$) приведено на рис. 5.50.

**Шаг 9**

К модифицированным матрицам $\Omega^{\wedge\wedge}_b$ применим операцию обратного ДКП (модуль (М.86)) и на основе воссозданных блоков формируем общий массив контейнера (модуль (М.87)).

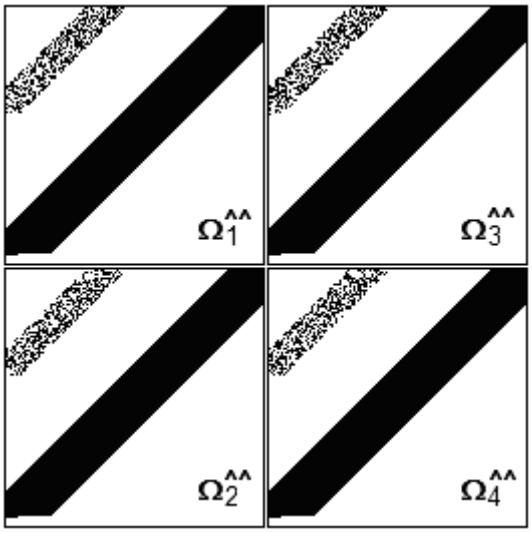



**Рис. 5.50. Массивы модифицированных (изображено черным) НЧ и СЧ-коэффициентов ДКП**

$$
\begin{aligned}
&\text{БE} := \left|\begin{array}{l}
\text{for}\ \ b \in 1 .. \aleph_C \\
\quad \text{БE}_b \leftarrow \zeta^T \cdot \Omega\hat{}\hat{}_b \cdot \zeta \\
\text{БE}
\end{array}\right. \qquad \text{(M.86)} \\
&S := \left|\begin{array}{l}
S \leftarrow \text{БE}_1 \\
\text{for}\ \ b \in 2 .. X \div N \\
\quad S \leftarrow \text{stack}\left(S, \text{БE}_b\right) \\
S' \leftarrow 0 \\
\text{for}\ \ b \in X \div N + 1 .. \aleph_C \\
\quad \left|\begin{array}{l}
S' \leftarrow \text{БE}_b \ \ \text{if}\ \ S' = 0 \\
S' \leftarrow \text{stack}\left(S', \text{БE}_b\right) \ \ \text{otherwise} \\
\text{if}\ \ \text{mod}(b, X \div N) = 0 \\
\quad \left|\begin{array}{l} S \leftarrow \text{augment}(S, S') \\ S' \leftarrow 0 \end{array}\right.
\end{array}\right. \\
S \leftarrow \dfrac{S + |\min(S)|}{\max(S + |\min(S)|)} \cdot 255
\end{array}\right. \qquad \text{(M.87)}
\end{aligned}
$$

Контейнер со встроенным ЦВЗ изображен на рис.5.51.

**Шаг 10**

При извлечении ЦВЗ контейнер $\mathbf{S^*} \approx \mathbf{S}$ предварительно разбивается на блоки размерностью $\mathbf{N^*}\times\mathbf{N^*}$, $\mathbf{N^*} := \mathbf{N}$, общим количеством $\aleph^*_C := \aleph_C$ — модуль (М.55) с соответствующей подстановкой указанных переменных.

Для каждого блока выполняется трансформация (5.37), для чего используется программный модуль, подобный модулю (М.75). Для параметра $\alpha^* = \alpha$ формируется прогрессия $\tau^*$ и определяется индексная функция $\mathbf{ind^*(t)}$.

S

**Рис. 5.51. Изображение со встроенным ЦВЗ при параметрах $\alpha$ = 0.1 и $\gamma$ = 1**

Выполняется операция прямого ДКП трансформированных блоков изображения (модуль (М.78)).

Для извлечения скрытых данных из матриц ДКП предварительно должны быть заданными (либо вычисленными по заранее оговоренным с передающей стороной алгоритмам) следующие параметры: $\aleph^*_W := \aleph_W$, $L^* := L$, $H^* := H$, $L^{*'} := L'$, $H^{*'} := H'$, $\aleph^*_{\Omega mid} := \aleph_{\Omega mid}$, $\xi^* := \xi$. Программный модуль извлечения (М.88) представлен ниже.

Первым циклом перебирания индексов блоков контейнера из полученных матриц ДКП выбираются те НЧ-коэффициенты, в которые было произведено встраивание данных. Указанные коэффициенты выступают в роли аргументов индексной функции, формируя вектор **low**. Полученный для каждого блока результат вычисления заносится в **b**-ый элемент массива $W^{*'}_{vec}$.

Следующим циклом перебирания индексов блоков из матриц ДКП выбираются модифицированные СЧ-коэффициенты, которые формируют вектор **mid**. Владея предварительной информацией о том, что ЦВЗ был разделен поровну между блоками и элементы ЦВЗ представлены целыми числами от 1 до 16, из элемента массива $\xi$, который имеет индекс $n + (b-1)\cdot \aleph^*_W / \aleph^*_C$, выделяются все 16 сегментов длиной по $\aleph^*_{\Omega mid}$ элементов.

Для каждого из этих сегментов вычисляется взаимная корреляционная функция с вектором **mid** выделенных СЧ-коэффициентов. Проводится поиск индекса, который отвечает наибольшему значению корреляции. Если значение этого индекса меньше или больше 8, делается вывод о том, что встроенный элемент ЦВЗ имел значение, соответственно, +1 или –1, которое и присваивается элементу вектора **V**.

Указанный вектор имеет размерность $(\aleph^*_W / \aleph^*_C) \times 1$ и формируется отдельно для каждого блока, образовывая впоследствии $(\aleph^*_C + 1)$-й элемент массива $W^{*'}_{vec}$.

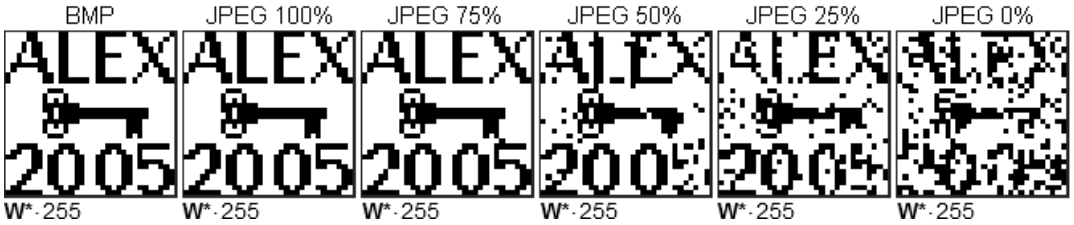
Таким образом, результирующий массив $W^{*'}_{vec}$ состоит из 5 элементов, каждый из которых, в свою очередь, является вектором размерностью $\aleph^*_W \times 1$. Элементы массива $W^{*'}_{vec}$, по сути, представляют собой 5 гипотез относительно возможного вида ЦВЗ, которые в идеале должны быть идентичными. Для формирования единственной гипотезы используется «принцип большинства» — **j**-му элементу результирующего вектора $W^*_{vec}$ присваивается такое значение, которое преобладает среди всех 5 элементов, имеющих индекс **j**.

$$
\begin{aligned}
&W^{*}_{vec} := \left|\begin{array}{l}
\textbf{for}\ \ b \in 1..\aleph^{*}_{C} \\
\quad \left|\begin{array}{l}
j \leftarrow 1 \\
\textbf{for}\ \ \upsilon \in 1..N^{*} \\
\quad \left|\begin{array}{l}
\textbf{break}\ \ \textbf{if}\ \ j > \aleph^{*}_{W} \\
\textbf{for}\ \ \nu \in 1..N^{*} \\
\quad \left|\begin{array}{l}
\textbf{if}\ \ L^{*} < (\upsilon + \nu) < H^{*} \\
\quad \left|\begin{array}{l} low_{j} \leftarrow ind\left[\left|\left(\Omega^{*}_{b}\right)_{\upsilon,\nu}\right|\right] \\ j \leftarrow j + 1 \end{array}\right. \\
\textbf{break}\ \ \textbf{if}\ \ j > \aleph^{*}_{W}
\end{array}\right. \\
W^{*'}vec_{b} \leftarrow low
\end{array}\right. \\
\textbf{for}\ \ b \in 1..\aleph^{*}_{C} \\
\quad \left|\begin{array}{l}
j \leftarrow 1 \\
\textbf{for}\ \ \upsilon \in 1..N^{*} \\
\quad \left|\begin{array}{l}
\textbf{break}\ \ \textbf{if}\ \ j > \aleph^{*}_{\Omega mid} \\
\textbf{for}\ \ \nu \in 1..N^{*} \\
\quad \left|\begin{array}{l}
\textbf{if}\ \ L^{**} < (\upsilon + \nu) < H^{**} \\
\quad \left|\begin{array}{l} mid_{j} \leftarrow \left(\Omega^{*}_{b}\right)_{\upsilon,\nu} \\ j \leftarrow j + 1 \end{array}\right. \\
\textbf{break}\ \ \textbf{if}\ \ j > \aleph^{*}_{\Omega mid}
\end{array}\right. \\
\textbf{for}\ \ n \in 1..\dfrac{\aleph^{*}_{W}}{\aleph^{*}_{C}} \\
\quad \left|\begin{array}{l}
\textbf{for}\ \ m \in 1..16 \\
\quad \left|\begin{array}{l}
\eta \leftarrow submatrix\left[\xi^{*}_{\,n+\frac{\aleph^{*}_{W}}{\aleph^{*}_{C}}\cdot(b-1)}, m, \aleph^{*}_{\Omega mid} + m - 1, 1, 1\right] \\
K_{m} \leftarrow \sum_{i=1}^{\aleph^{*}_{\Omega mid}} mid_{i} \cdot \eta_{i}
\end{array}\right. \\
\textbf{for}\ \ m \in 1..16 \\
\quad MAX \leftarrow m \ \ \textbf{if}\ \ K_{m} = \max(K) \\
V_{n} \leftarrow 1 \ \ \textbf{if}\ \ MAX < 8 \\
V_{n} \leftarrow -1 \ \ \textbf{if}\ \ MAX > 8
\end{array}\right. \\
W^{*'}vec_{\aleph^{*}_{C}+1} \leftarrow V \ \ \textbf{if}\ \ b = 1 \\
W^{*'}vec_{\aleph^{*}_{C}+1} \leftarrow stack\left(W^{*'}vec_{\aleph^{*}_{C}+1}, V\right) \ \ \textbf{if}\ \ b > 1
\end{array}\right. \\
\textbf{for}\ \ j \in 1..\aleph^{*}_{W} \\
\quad \left|\begin{array}{l}
\textbf{for}\ \ b \in 1..\aleph^{*}_{C} + 1 \\
\quad \omega_{b} \leftarrow \left(W^{*'}vec_{b}\right)_{j} \\
W^{*}vec_{j} \leftarrow 1 \ \ \textbf{if}\ \ mean(\omega) > 0 \\
W^{*}vec_{j} \leftarrow -1 \ \ \textbf{if}\ \ mean(\omega) < 0
\end{array}\right. \\
W^{*}_{vec}
\end{array}\right.
\end{aligned}
\qquad (M.88)
$$

Для случая, рассмотренного в модуле (М.88), из этих 5 элементов формируется вектор $\omega$, для которого подсчитывается среднее значение. Если последнее окажется положительным (количество единиц преобладает), то элемент $W^*_{vec_j} = 1$. В противном случае — $W^*_{vec_j} = -1$.

Полученный вектор $W^*_{vec}$ сворачивается в массив $W^*$ с размерностью оригинального ЦВЗ (за основу можно использовать программный модуль (М.74)).

На рис. 5.52 представлен ЦВЗ, извлеченный из контейнера, который предварительно был подвергнут JPEG-компрессии. В первых двух случаях очевидно полное соответствие извлеченного ЦВЗ оригиналу. Количество ошибочно распознанных элементов ЦВЗ в третьем случае составило 3 пикселя (или 0,29% от общего их количества $\aleph^*_W$); для *г)* — 58 (5,66%); для *д)* — 109 (10,64%); для *е)* — 194 (18,95%).




*а)* *б)* *в)* *г)* *д)* *е)*

**Рис. 5.52. ЦВЗ (увеличено), извлеченные из контейнера $S^*$ ($\alpha = 0.1$, $\gamma = 1$): формат $S^*$ не изменен (*а*); применена JPEG-компрессия с сохранением 100% качества изображения (*б*); то же при сохранении 75, 50, 25 и 0% качества (соответственно: *в*, *г*, *д*, *е*)**

Увеличение параметров $\alpha$ и, особенно, $\gamma$ делает данную стеганосистему еще более устойчивой к атаке компрессией, однако при этом сильно страдает качество изображения.

Результаты вычисления показателей визуального искажения контейнера при значениях параметров $\alpha = 0.1$, $\gamma = 1$ внесены в табл.5.4. При этом следует иметь в виду, что объем изображения-контейнера был увеличен в четыре раза, по сравнению с контейнерами, используемыми при рассмотрении предыдущих методов.

### *5.3.4. Методы расширения спектра*

Изначально методы расширения спектра (РС или SS — *Spread-Spectrum*) использовались при разработке военных систем управления и связи. Во время Второй мировой войны в радиолокации расширение спектра использовалось для борьбы с намеренными помехами. В последние годы развитие данной технологии объясняется желанием создать эффективные системы радиосвязи для обеспечения высокой помехоустойчивости при передаче узкополосных сигналов по каналам с шумами и осложнения их перехвата. Система связи является системой с расширенным спектром в следующих случаях [65]:

**1.** Полоса частот, которая используется при передаче, значительно шире минимально необходимой для передачи текущей информации. При этом энергия информационного сигнала расширяется на всю ширину полосы частот при низком соотношении сигнал/шум, в результате чего сигнал трудно обнаружить, перехватить или воспрепятствовать его передаче путем внесения помех. Хотя сум-

Таблица 5.4.

**Показатели визуального искажения в случае скрытия данных в частотной области изображения**

| *Название показателя искажения* | *Оригинал* | *Методы скрытия в частотной области* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *Коха- Жао* (**P** = 0.5) | *Коха-Жао* (**P** = 25) | *Бенгама-Мемона* (**P** = 1) | *Бенгама-Мемона* (**P** = 35) | *Хсу-Ву (а)* | *Хсу-Ву (б)* | *Фридрих* ($\alpha$ = 0.1, $\gamma$ = 1) | *Фридрих* ($\alpha$ = 0.1, $\gamma$ = 2) | *Фридрих* ($\alpha$ = 0.2, $\gamma$ = 1) |
| Максимальная разность, ***MD*** | 0 | 39 | 45 | 45 | 51 | 122 | 88 | 68 | 94 | 73 |
| Средняя абсолютная разность, ***AD*** | 0 | 9.504 | 11.392 | 1.846 | 3.042 | 36.805 | 26.457 | 19.233 | 25.990 | 22.413 |
| Нормированная средняя абсолютная разность, ***NAD*** | 0 | 0.074 | 0.088 | 0.014 | 0.024 | 0.251 | 0.181 | 0.132 | 0.179 | 0.154 |
| Среднеквадратическая ошибка, ***MSE*** | 0 | 124.383 | 178.342 | 15.427 | 31.417 | 1794.673 | 962.154 | 543.072 | 942.113 | 718.449 |
| Нормированная среднеквадратическая ошибка, ***NMSE*** | 0 | $5.065 \cdot 10^{-3}$ | $7.263 \cdot 10^{-3}$ | $6.283 \cdot 10^{-4}$ | $1.279 \cdot 10^{-3}$ | 0.062 | 0.033 | 0.019 | 0.033 | 0.025 |
| $L^p$-норма, $p = 2$ | 0 | 11.153 | 13.354 | 3.928 | 5.605 | 42.364 | 31.019 | 23.304 | 30.694 | 26.804 |
| Лапласова среднеквадратическая ошибка, ***LMSE*** | 0 | 0.020 | 0.037 | 0.027 | 0.053 | 0.444 | 0.297 | 0.352 | 1.010 | 0.362 |
| Отношение “сигнал/шум”, ***SNR*** | $\infty$ | 197.423 | 137.690 | $1.592 \cdot 10^{3}$ | 781.605 | 16.056 | 29.948 | 52.178 | 30.078 | 39.441 |
| Максимальное отношение “сигнал/шум”, ***PSNR*** | $\infty$ | 522.782 | 364.608 | $4.215 \cdot 10^{3}$ | $2.070 \cdot 10^{3}$ | 36.232 | 67.583 | 119.735 | 69.020 | 90.507 |
| Качество изображения, ***IF*** | 1 | 0.994935 | 0.992737 | 0.999372 | 0.998721 | 0.937717 | 0.966609 | 0.980835 | 0.966753 | 0.974646 |
| Нормированная взаимная корреляция, ***NC*** | 1 | 0.993261 | 0.986178 | 0.996790 | 0.994154 | 0.788233 | 0.863134 | 0.903486 | 0.873017 | 0.876281 |
| Качество корреляции, ***CQ*** | 190.182 | 188.901 | 187.554 | 189.572 | 189.071 | 155.023 | 169.754 | 176.042 | 170.106 | 170.742 |
| Структурное содержание, ***SC*** | 1 | 1.008484 | 1.020804 | 1.005826 | 1.010523 | 1.565560 | 1.316381 | 1.210451 | 1.283235 | 1.285486 |
| Общее сигма-отношение “сигнал/шум”, ***GSSNR*** | $\infty$ | 87.792 | 60.244 | $4.388 \cdot 10^{3}$ | $2.167 \cdot 10^{3}$ | 6.326 | 74.183 | 18.153 | 10.080 | 13.899 |
| Сигма-отношение “сигнал/шум”, ***SSNR*** | $\infty$ | 72.9 | 67.3 | 73.3 | 69.2 | 30.1 | 40.8 | 198.8 | 158.1 | 178.9 |
| Нормированное отношение “сигма/ошибка”, ***NSER*** | 256 | 100 | 99 | 127 | 116 | 114 | 41 | 184 | 233 | 190 |
| Подобие гистограмм, ***HS*** | 0 | 11096 | 10714 | 2446 | 2736 | 15960 | 12712 | 43062 | 50984 | 45806 |

марная мощность сигнала может быть большой, соотношение сигнал/шум в любом диапазоне частот является малым, что делает сигнал с расширенным спектром трудно определяемым при радиосвязи и, в контексте скрытия информации стеганографическими методами, трудно различимым человеком.

2. Расширение спектра выполняется с помощью так называемого расширяющего (или кодового) сигнала, который не зависит от передаваемой информации. Присутствие энергии сигнала во всех частотных диапазонах делает радиосигнал с расширенным спектром стойким к внесению помех, а информацию, встроенную в контейнер методом расширения спектра, стойкой к ее устранению или извлечению из контейнера. Компрессия и другие виды атак на систему связи могут устранить энергию сигнала из некоторых участков спектра, но поскольку последняя была распространена по всему диапазону, в других полосах остается достаточное количество данных для восстановления информации. В результате, если, разумеется, не разглашать ключ, который использовался для генерации кодового сигнала, вероятность извлечения информации неавторизованными лицами существенно снижается.
3. Восстановление первичной информации (то есть «сужение спектра») осуществляется путем сопоставления полученного сигнала и синхронизированной копии кодового сигнала.

В радиосвязи применяют три основных способа расширения спектра:

- с помощью прямой ПСП (РСПП);
- с помощью скачкообразного перестраивания частот;
- с помощью компрессии с использованием линейной частотной модуляции (ЛЧМ).

При расширении спектра прямой последовательностью информационный сигнал модулируется функцией, которая принимает псевдослучайные значения в установленных пределах, и умножается на временну*ю* константу — частоту (скорость) следования элементарных посылок (элементов сигнала). Данный псевдослучайный сигнал содержит составляющие на всех частотах, которые, при их расширении, модулируют энергию сигнала в широком диапазоне частот. В методе расширения спектра с помощью скачкообразного перестраивания частот передатчик мгновенно изменяет одну частоту несущего сигнала на другую. Секретным ключом при этом является псевдослучайный закон изменения частот. При компрессии с использованием ЛЧМ сигнал модулируется функцией, частота которой изменяется во времени. Очевидно, что любой из указанных методов может быть распространен на использование в пространственной области при построении стеганографических систем.

Рассмотрим один из вариантов реализации метода РСПП, авторами которого являются Смит (J.R. Smith) и Комиски (B.O. Comiskey) [88]. Алгоритм модуляции следующий: каждый бит сообщения $\boldsymbol{m_i}$ представляется некоторой базисной функцией $\varphi_i$ размерностью $\boldsymbol{X}\times\boldsymbol{Y}$, умноженной, в зависимости от значения бита (1 или 0), на +1 или –1:

$$\boldsymbol{E}(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{i}} \boldsymbol{m}_{\boldsymbol{i}} \cdot \varphi_{\boldsymbol{i}}(\boldsymbol{x},\boldsymbol{y}) . \qquad (5.47)$$

Модулированное сообщение $\boldsymbol{E}(\boldsymbol{x},\boldsymbol{y})$, полученное при этом, попиксельно суммируется с изображением-контейнером $\boldsymbol{C}(\boldsymbol{x},\boldsymbol{y})$, в качестве которого используется полутоновое изображение размером $\boldsymbol{X}\times\boldsymbol{Y}$. Результатом является стеганоизображение $\boldsymbol{S}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{C}(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{E}(\boldsymbol{x},\boldsymbol{y})$, при $\boldsymbol{x}\in 1...\boldsymbol{X}$, $\boldsymbol{y}\in 1...\boldsymbol{Y}$.

Чтобы сделать невозможным искажение уже встроенного бита сообщения, базисные функции должны быть ортогональными:

$$\left\langle \varphi_{\boldsymbol{i}}, \varphi_{\boldsymbol{j}} \right\rangle = \sum_{\boldsymbol{x},\boldsymbol{y}}^{\boldsymbol{X},\boldsymbol{Y}} \varphi_{\boldsymbol{i}}(\boldsymbol{x},\boldsymbol{y}) \cdot \varphi_{\boldsymbol{j}}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{n}_{\varphi} \cdot \boldsymbol{G}^2 \cdot \delta_{\boldsymbol{i},\boldsymbol{j}}, \qquad (5.48)$$

где $n_\varphi$ — количество значащих пикселей в базисной функции; $G^2$ — средняя мощность, приходящаяся на пиксель, $n_\varphi \cdot G^2 = \sum_{x,y} \varphi_i^2(x,y)$; $\delta_{i,j} = \begin{cases} 1, \text{ при } i = j; \\ 0, \text{ при } i \neq j \end{cases}$ – дельта-символ Кронекера.

В идеальном случае, все базисные функции $\varphi_i$ должны быть некоррелированными с изображением-контейнером $C$, то есть должны быть ортогональными к нему: $\langle C, \varphi_i \rangle = 0, \ \forall i$. Однако на практике тяжело найти контейнер, который был бы полностью ортогональным ко всем базисным функциям $\varphi_i$. В таком случае должна быть введена величина погрешности $\langle C, \varphi_i \rangle = \Delta \approx 0$, которая учитывается увеличением мощности $G^2$.

Для эффективного скрытия информации необходимо значительное количество базисных функций, ортогональных к типичным изображениям. Кодирование же изображений выдвигает противоположное требование: идеальным считается небольшое количество базисных функций, которые приблизительно перекрывают всю область изображения. Эти требования вступают в конфликт, когда изображение, содержащее скрытую информацию, подвергается компрессии: идеальная схема компрессии не способна полностью отобразить базисы, которые использовались для скрытия.

Базисные функции могут быть организованы и сравнимы в соответствии с такими свойствами как полная мощность, степень пространственного расширения (или локализации), а также степень пространственного частотного расширения (или локализации) [88].

В случае РСПП моделирующая функция состоит из постоянного коэффициента усиления $G$ (целое число), умноженного на псевдослучайный блок (массив) базисных функций $\varphi_i$ значений $\pm 1$. Каждый массив $\varphi_i$ имеет индивидуальное расположение в ($x$, $y$)-массиве. Кроме того, массивы $\varphi_i$ являются непересекающимися (то есть заведомо ортогональными друг к другу) и перекрывают ($x$, $y$)-массив без промежутков.

Также будем считать, что все $N_\varphi$ базисные функции имеют одинаковое количество значащих элементов ($n_\varphi$). В этом случае полную мощность можно записать следующим образом:

$$P \equiv \sum_{x,y}^{X,Y} \left( \sum_i G \cdot m_i \cdot \varphi_i(x,y) \right)^2 = \sum_i \sum_{x,y}^{X,Y} \left( G \cdot m_i \cdot \varphi_i(x,y) \right)^2 = G^2 \cdot X \cdot Y = N_\varphi \cdot n_\varphi \cdot G^2 . \quad (5.49)$$

На этапе извлечения информации нет необходимости владеть информацией о первичном контейнере $C$. Операция декодирования заключается в восстановлении скрытого сообщения путем проецирования полученного стеганоизображения $S^*$ на все базисные функции $\varphi_i$:

$$\sigma_i = \langle S^*, \varphi_i \rangle = m_i \cdot n_\varphi \cdot G^2 .$$

Значения $m_i$ могут быть легко восстановлены с помощью знаковой функции:

$$m_i^* = sign(\sigma_i) = \begin{cases} -1, \text{ при } \sigma_i < 0; \\ 1, \text{ при } \sigma_i > 0; \\ ? \text{ при } \sigma_i = 0, \end{cases} \quad \text{при условии, что } G^2 >> 0 . \quad (5.50)$$

Если $\sigma_i = 0$, то скрываемая информация была утрачена. При малых значениях средней мощности $G^2$ возрастает вероятность извлечения ошибочного значения бита информации, однако информация при этом искажается в меньшей степени.

Основное преимущество стеганографических методов основанных на расширении спектра — сравнительно высокая стойкость к различного рода атакам на изображение, поскольку скрываемая информация распределена в широкой полосе частот и ее трудно удалить без полного разрушения контейнера. Искажения стеганоизображения увеличивают значение $\Delta$, однако до тех пор, пока выполняется условие $|\Delta| < |n_\varphi \cdot G^2|$, скрытое сообщение не пострадает.

Приведем пример реализации метода стеганографического скрытия с помощью расширения спектра в программе MathCAD.

**Шаг 1**

Импортируем изображение-контейнер:

**C** := **READBMP**("C.bmp");

**X** := **rows**(**C**); **X** = 128;

**Y** := **cols**(**C**); **Y** := 128.

**Шаг 2**

Формируем массив **Φ** ортогональных базисных функций, который должен иметь размерность сигнала-контейнера (**X**×**Y**) и представлять собой сумму всех базисных не перекрывающихся функций $\phi_i$.

Очевидно, что для получения ортогональных базисных функций $\phi_i$, достаточно провести деление массива **Φ** на непересекающиеся сегменты, каждый из которых поместить в массив нулевых элементов размерностью **X**×**Y** по соответствующим координатам.

Генерирование массива **Φ** осуществляется программным модулем (М.89). При этом +1 и –1 чередуются в шахматном порядке. Конечно, при создании более надежной стеганосистемы необходимо выбрать более сложный алгоритм формирования массива ортогональных базисных функций.

```
Φ := | d ← −1
     | for x ∈ 1 .. X
     |   | d ← −d
     |   | for y ∈ 1 .. Y
     |   |   | Φ[x,y] ← d
     |   |   | d ← −d
     | Φ
```

(М.89)

**Шаг 3**

Пусть общее количество базисных функций $N_\phi$ := 256. Размерность значащего подмассива отдельной базисной функции (размерность выделяемого сегмента) определим, исходя из размерности массива **Φ** и общего количества базисных функций $N_\phi$:

$$n := \mathrm{floor}\left(\sqrt{\frac{X \cdot Y}{N_\phi}}\right); \quad n = 8.$$

Формирование массива базисных функций $\phi$ осуществим путем выделения соответствующего базисной функции значащего подмассива $\mathbf{n}\times\mathbf{n}$ из общего массива $\Phi$ и последующего его встраивания на соответствующие позиции нулевого массива размерностью $\mathbf{X}\times\mathbf{Y}$ с помощью функции **putregion**(•). Указанные операции осуществляет программный модуль (М.90).

```
ϕ := | ℑ ← Vrnd(s)
     | c1 ← 1
     | c2 ← n
     | for i ∈ 1 .. N_ϕ
     |   | r1 ← mod[n·(i − 1) + 1, X]
     |   | r2 ← r1 + n − 1
     |   | ϕ_(ℑ_i) ← putregion(0·Φ, submatrix(Φ, r1, r2, c1, c2), r1, c1)
     |   | if r2 = X
     |   |   | c1 ← c1 + n
     |   |   | c2 ← c2 + n
     | ϕ
```

(М.90)

Для упрощения, нулевой массив формируется умножением на 0 массива $\Phi$. Результат присваивается псевдослучайному элементу массива $\phi$. Для генерирования ПСП используется программный модуль (М.57) при следующих исходных значениях:

- примитивный полином 8-й степени ($\mathbf{d} := \mathbf{log}(\mathbf{N}_\phi, 2)$, $\mathbf{d} = 8$), например, следующего вида: $\rho(x) = 1 + x^3 + x^5 + x^7 + x^8$, что означает
$$\mu \leftarrow (0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 1)^{\mathrm{T}};$$
- значение начального состояния регистра $\mathbf{s}$ — произвольное целое число из диапазона возможных значений $[1; \mathbf{N}_\phi)$.

Графическое отображение массива $\Phi$ и некоторых массивов базисных функций $\phi_i$ приведены на рис. 5.53.

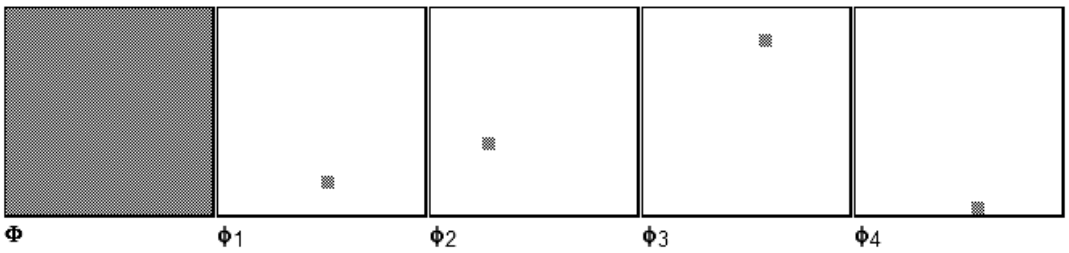



**Рис. 5.53. Графическая интерпретация массивов ортогональных базисных функций**

**Шаг 4**

Рассмотрим степень ортогональности сигнала контейнера $\mathbf{C}$ к полученным базисным функциям $\phi_i$, для чего воспользуемся программным модулем (М.91).

Результат вычисления модуля (М.91) изображен на рис. 5.54. При выбранном контейнере и алгоритме формирования массива $\Phi$ максимальное абсолютное значение погрешности ортогональности $\Delta$ составляет 312.

$$\Delta := \left| \begin{array}{l} \text{for } i \in 1..N_{\phi} \\ \quad \Delta_i \leftarrow \sum_{x=1}^{X} \sum_{y=1}^{Y} (\phi_i)_{x,y} \cdot C_{x,y} \\ \Delta \end{array} \right. \qquad \text{(M.91)}$$

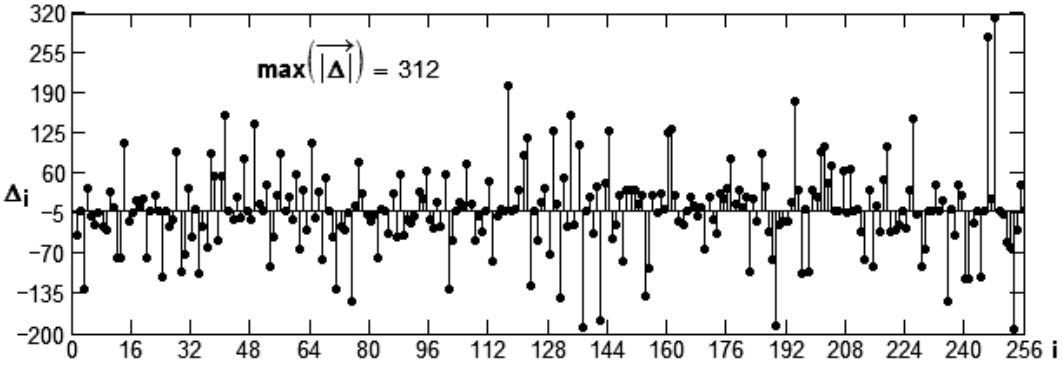



**Рис. 5.54. График степени ортогональности массива контейнера C к базисным функциям $\phi_i$**

**Шаг 5**

Пусть скрываемое сообщение — **M** := "© Пузыренко А.Ю., 2005 г.". Двоичный объем сообщения составляет $L_M = 200$ бит, что не превышает общего количества базисных функций $N_\phi = 256$.

Используя программный модуль (М.92), проведем модуляцию сообщения **M** базисными функциями (см. формулу (5.47)), предварительно присвоив тем элементам двоичного вектора сообщения, которые имели нулевое значение, значение –1.

$$E := \left| \begin{array}{l} \text{Ⓘ -- см. (М.21), кроме } S \leftarrow C \\ \text{for } j \in 1..L_M \\ \quad M_{vec\_bin_j} \leftarrow -1 \text{ if } M_{vec\_bin_j} = 0 \\ (*)\left\{ \begin{array}{l} \text{for } x \in 1..X \\ \quad \text{for } y \in 1..Y \\ \qquad E_{x,y} \leftarrow \sum_{i=1}^{L_M} M_{vec\_bin_i} \cdot (\phi_i)_{x,y} \end{array} \right. \\ E \end{array} \right. \qquad \text{(M.92)}$$

Результат модуляции приведен на рис. 5.55.

Присутствие на рис. 5.55 полностью черных сегментов $n \times n$ говорит о том, что некоторые базисные функции не принимали участия в модуляции по причине отсутствия в сообщении бит с соответствующими им индексами. В данном случае количество неиспользованных базисных функций:

$$N_\phi - L_M = 56.$$

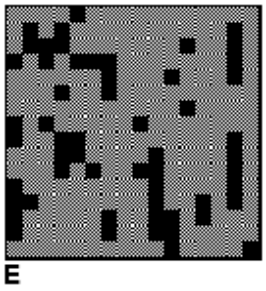



**Рис. 5.55. Пример модулированного сообщения**

**Шаг 6**

С учетом максимального значения погрешности $\Delta$, проведем вычисление достаточного коэффициента усиления мощности встроенного в контейнер модулированного сообщения. Для этого используем специальный вычислительный модуль решения неравенств (М.93).

$$\begin{aligned} &K_G := 1; \\ &n_\phi := n^2; \\ &\text{Given} \quad n_\phi \cdot K_G > \max\left(\overrightarrow{|\Delta|}\right); \\ &\qquad K_G := \text{ceil}\left(\text{Find}\left(K_G\right)\right),\ K_G = 5. \end{aligned} \tag{М.93}$$

Данный модуль открывается директивой **Given**, после которой следует логическое неравенство, в выполнении которого мы заинтересованы. Функция **Find**($K_G$) возвращает минимальное значение переменной $K_G$, предварительно заданной как $K_G := 1$, позволяющее получить точное решение неравенства. Полученный результат с помощью функции **ceil**(•) округляется до наименьшего целого, превышающего точное решение.

**Шаг 7**

Принимая во внимание значение коэффициента $K_G$, проводим предварительное нормирование массива контейнера, используя программный модуль (М.94).

$$C_{norm} := \left| \begin{array}{l} \text{for } x \in 1..X \\ \quad \text{for } y \in 1..Y \\ \quad\quad C_{norm_{x,y}} \leftarrow \text{round}\left[\dfrac{C_{x,y}}{255}\cdot\left(255 - 2\cdot K_G\right) + K_G\right] \\ C_{norm} \end{array} \right. \tag{М.94}$$

Без проведения такой операции весьма вероятны случаи, когда значения яркостей отдельных пикселей выйдет за пределы диапазона [0, 255]. При этом яркость $\lambda$ будет вычисляться с помощью функции записи в файл изображения как значение $\lambda' = \text{mod}[\text{mod}(\lambda, 256) + 256, 256]$. А это вызовет ощутимый зрением человека скачок значения яркости через весь диапазон ее изменения и, очевидно, является недопустимым[12].

До нормирования имеем: $\min(C) = 0$, $\max(C) = 255$. После выполнения модуля (М.94): $\min(C_{norm}) = 5$, $\max(C_{norm}) = 250$.

Очевидно, что даже в случае прибавления к граничным значениям яркости пикселя контейнера $C_{norm_{x,y}}$ элемента, модулированного определенной базисной функцией сообщения (который может принимать значение $\pm K_G \cdot E_{x,y}$), яркость пикселя заполненного контейнера $S_{x,y}$ не выйдет за допустимые пределы (рис. 5.56):

$$S := C_{norm} + K_G \cdot E; \quad \text{при этом} \quad \min(S) = 0, \quad \max(S) = 255.$$

**Шаг 8**

Для извлечения сообщения должны быть известны:

- стеганоконтейнер **S***;

[12] Например, при $\lambda = -8$ $\lambda' = 248$; при $\lambda = 259$ $\lambda' = 3$.

- размерность контейнера **X*** и **Y***;
- общее количество базисных функций $\mathbf{N}^*_{\phi}$;
- конфигурация (в простейшем случае — **n***) и алгоритм получения базисных функций **ϕ***.

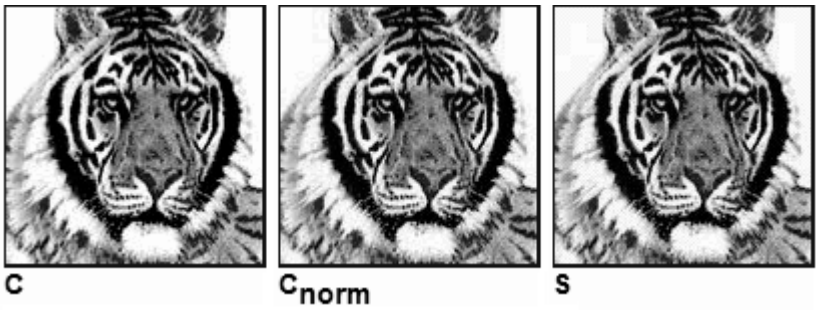



**Рис. 5.56. Пустой (C), нормированный ($C_{norm}$) и заполненный при $K_G = 5$ (S) контейнеры**

Программный модуль извлечения скрытого путем расширения спектра прямой последовательностью сообщения — (М.95).

Результат вычисления показателей искажения контейнера сведено в табл. 5.5.

$$
\begin{aligned}
&\mathbf{M}^* := \Big| \ \textbf{for}\ \ i \in 1 .. N^*_{\phi} \\
&\quad \left| \begin{array}{l} \mu \leftarrow \sum_{x=1}^{X^*} \sum_{y=1}^{Y^*} S^*_{x,y} \cdot \left(\phi^*_i\right)_{x,y} \\ M^*_{vec\_bin_i} \leftarrow 1 \ \ \textbf{if}\ \ \mu > 0 \\ M^*_{vec\_bin_i} \leftarrow 0 \ \ \textbf{if}\ \ \mu < 0 \\ M^*_{vec\_bin_i} \leftarrow \text{round}(\text{rnd}(1)) \ \ \textbf{if}\ \ \mu = 0 \end{array} \right\} (*) \\
&\quad \textbf{for}\ \ j \in 1 .. \text{rows}\left(M^*_{vec\_bin}\right) \div 8 \\
&\qquad M^*_{vec_j} \leftarrow \text{B2D}\left(\text{submatrix}\left(M^*_{vec\_bin}, 8 \cdot j - 7, 8 \cdot j, 1, 1\right)\right) \\
&\quad M^*_{vec}
\end{aligned}
\qquad \text{(М.95)}
$$

Более эффективным, по мнению авторов [88], является алгоритм реализации метода РСПП, который заключается в использовании «двойного канала» встраивания: каждая базисная функция $\phi_i$ по заранее оговоренному алгоритму делится на две равные значащие части, которые модулируются последовательностями, соответственно, (–1)(+1) для встраивания «0», и (+1)(–1) для встраивания «1». При извлечении сообщения из контейнера, каждый бит является результатом двойной демодуляции — для случаев (–1)(+1) и (+1)(–1) проводится вычисление значений функции корреляции. Очевидно, что истинное (при идеальном канале связи) значение бита сообщения («0» или «1») будет определяться б*о*льшим значением корреляции.

Реализация приведенного алгоритма требует внесения изменений в блоки «(*)» программных модулей встраивания (М.92) и извлечения (М.95). Возможный вариант замены представлен фрагментами программных модулей (М.96), (М.97).

Как на этапе встраивания, так и во время извлечения, для деления значащих элементов базисных функций на два подмножества используется значение переменной-счетчика **V** (в первом случае (модуль (М.96)) — как вектора).

$$
\begin{array}{l}
\mathbf{E} \leftarrow \mathbf{C} \cdot 0 \\
\mathbf{V} \leftarrow \mathbf{M}_{\mathbf{vec\_bin}} \cdot 0 \\
\textbf{for}\ \ \mathbf{x} \in 1 .. \mathbf{X} \\
\quad \textbf{for}\ \ \mathbf{y} \in 1 .. \mathbf{Y} \\
\quad\quad \textbf{for}\ \ \mathbf{j} \in 1 .. \mathbf{L}_{\mathbf{M}} \\
\quad\quad\quad \textbf{if}\ \ \mathbf{M}_{\mathbf{vec\_bin}_{\mathbf{j}}} = -1 \\
\quad\quad\quad\quad \mathbf{E}_{\mathbf{x},\mathbf{y}} \leftarrow \mathbf{E}_{\mathbf{x},\mathbf{y}} - \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \ \ \textbf{if}\ \ \mathbf{V}_{\mathbf{j}} < \dfrac{\mathbf{n}^2}{2} \\
\quad\quad\quad\quad \mathbf{E}_{\mathbf{x},\mathbf{y}} \leftarrow \mathbf{E}_{\mathbf{x},\mathbf{y}} + \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \ \ \textbf{if}\ \ \mathbf{V}_{\mathbf{j}} \geq \dfrac{\mathbf{n}^2}{2} \\
\quad\quad\quad\quad \mathbf{V}_{\mathbf{j}} \leftarrow \mathbf{V}_{\mathbf{j}} + 1 \ \ \textbf{if}\ \ \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \neq 0 \\
\quad\quad\quad \textbf{if}\ \ \mathbf{M}_{\mathbf{vec\_bin}_{\mathbf{j}}} = 1 \\
\quad\quad\quad\quad \mathbf{E}_{\mathbf{x},\mathbf{y}} \leftarrow \mathbf{E}_{\mathbf{x},\mathbf{y}} + \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \ \ \textbf{if}\ \ \mathbf{V}_{\mathbf{j}} < \dfrac{\mathbf{n}^2}{2} \\
\quad\quad\quad\quad \mathbf{E}_{\mathbf{x},\mathbf{y}} \leftarrow \mathbf{E}_{\mathbf{x},\mathbf{y}} - \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \ \ \textbf{if}\ \ \mathbf{V}_{\mathbf{j}} \geq \dfrac{\mathbf{n}^2}{2} \\
\quad\quad\quad\quad \mathbf{V}_{\mathbf{j}} \leftarrow \mathbf{V}_{\mathbf{j}} + 1 \ \ \textbf{if}\ \ \left(\boldsymbol{\phi}_{\mathbf{j}}\right)_{\mathbf{x},\mathbf{y}} \neq 0
\end{array}
\tag{M.96}
$$

$$
\begin{array}{l}
\mu_{--} \leftarrow 0 \\
\mu_{+} \leftarrow 0 \\
\mathbf{V} \leftarrow 0 \\
\textbf{for}\ \ \mathbf{x} \in 1 .. \mathbf{X}^* \\
\quad \textbf{for}\ \ \mathbf{y} \in 1 .. \mathbf{Y}^* \\
\quad\quad \textbf{if}\ \ \mathbf{V} < \dfrac{\mathbf{n}^{*2}}{2} \\
\quad\quad\quad \mu_{--} \leftarrow \mu_{--} - \mathbf{S}^*_{\mathbf{x},\mathbf{y}} \cdot \left(\boldsymbol{\phi}^*_{\mathbf{i}}\right)_{\mathbf{x},\mathbf{y}} \\
\quad\quad\quad \mu_{+} \leftarrow \mu_{+} + \mathbf{S}^*_{\mathbf{x},\mathbf{y}} \cdot \left(\boldsymbol{\phi}^*_{\mathbf{i}}\right)_{\mathbf{x},\mathbf{y}} \\
\quad\quad \textbf{if}\ \ \mathbf{V} \geq \dfrac{\mathbf{n}^{*2}}{2} \\
\quad\quad\quad \mu_{--} \leftarrow \mu_{--} + \mathbf{S}^*_{\mathbf{x},\mathbf{y}} \cdot \left(\boldsymbol{\phi}^*_{\mathbf{i}}\right)_{\mathbf{x},\mathbf{y}} \\
\quad\quad\quad \mu_{+} \leftarrow \mu_{+} - \mathbf{S}^*_{\mathbf{x},\mathbf{y}} \cdot \left(\boldsymbol{\phi}^*_{\mathbf{i}}\right)_{\mathbf{x},\mathbf{y}} \\
\quad\quad \mathbf{V} \leftarrow \mathbf{V} + 1 \ \ \textbf{if}\ \ \left(\boldsymbol{\phi}^*_{\mathbf{i}}\right)_{\mathbf{x},\mathbf{y}} \neq 0 \\
\mathbf{M}^*_{\mathbf{vec\_bin}_{\mathbf{i}}} \leftarrow 1 \ \ \textbf{if}\ \ \mu_{+} > \mu_{--} \\
\mathbf{M}^*_{\mathbf{vec\_bin}_{\mathbf{i}}} \leftarrow 0 \ \ \textbf{if}\ \ \mu_{+} < \mu_{--} \\
\mathbf{M}^*_{\mathbf{vec\_bin}_{\mathbf{i}}} \leftarrow \mathbf{round}(\mathbf{rnd}(1)) \ \ \textbf{if}\ \ \mu_{--} = \mu_{+}
\end{array}
\tag{M.97}
$$

Значения показателей искаженности контейнера в случае использования данного алгоритма занесены в табл. 5.5.

Таблица 5.5.

**Показатели визуального искажения в случае скрытия данных методом расширения спектра**

| *Название показателя искажения* | *Оригинал* | *РСПП-1* | *РСПП-2* |
|---|---|---|---|
| Максимальная разность, ***MD*** | 0 | 10 | 10 |
| Средняя абсолютная разность, ***AD*** | 0 | 4.614 | 4.624 |
| Нормированная средняя абсолютная разность, ***NAD*** | 0 | 0.031 | 0.032 |
| Среднеквадратическая ошибка, ***MSE*** | 0 | 31.721 | 31.817 |
| Нормированная среднеквадратическая ошибка, ***NMSE*** | 0 | $1.101 \cdot 10^{-3}$ | $1.104 \cdot 10^{-3}$ |
| $L^p$-норма, $p = 2$ | 0 | 5.632 | 5.641 |
| Лапласова среднеквадратическая ошибка, ***LMSE*** | 0 | 0.113 | 0.102 |
| Отношение “сигнал/шум”, ***SNR*** | $\infty$ | 908.381 | 905.628 |
| Максимальное отношение “сигнал/шум”, ***PSNR*** | $\infty$ | $2.050 \cdot 10^{3}$ | $2.044 \cdot 10^{3}$ |
| Качество изображения, ***IF*** | 1 | 0.998899 | 0.998896 |
| Нормированная взаимная корреляция, ***NC*** | 1 | 0.986079 | 0.986052 |
| Качество корреляции, ***CQ*** | 196.672 | 193.934 | 193.929 |
| Структурное содержание, ***SC*** | 1 | 1.027477 | 1.027529 |
| Общее сигма-отношение “сигнал/шум”, ***GSSNR*** | $\infty$ | 568.540 | 563.058 |
| Сигма-отношение “сигнал/шум”, ***SSNR*** | $\infty$ | 103.8 | 103.5 |
| Нормированное отношение “сигма/ошибка”, ***NSER*** | 256 | 62 | 61 |
| Подобие гистограмм, ***HS*** | 0 | 5702 | 5736 |

### *5.3.5. Другие методы скрытия данных в неподвижных изображениях*

#### 5.3.5.1. Статистические методы

В основу статистических методов скрытия конфиденциальных данных положена модификация определенных статистических свойств изображения (или же его фрагментов) с последующей проверкой статистических гипотез во время извлечения или проверки наличия указанных данных в контейнере. Сущность статистических методов сводится к такой модификации некоторых статистических характеристик контейнера, при которой принимающая сторона будет иметь возможность распознавать пустой контейнер от заполненного.

Как и в вышерассмотренных методах, многоразрядную статистическую стеганосистему можно получить путем разбивания контейнера на достаточное количество непересекающихся блоков (в общем случае это количество равняется количеству бит $l_M$ в скрываемом сообщении): $b_1, ..., b_{l_M}$ . При этом, отдельный бит сообщения $M_i$ встраивается в $i$-й блок контейнера. Детектирование скрытого в блоке бита выполняется путем использования так называемой проверочной (тестовой) функции. Последняя позволяет распознавать наличие модификации блока:

$$f(b_i) = \begin{cases} 1, & \text{блок } b_i \text{ был модифицирован;} \\ 0, & \text{блок } b_i \text{ не был модифицирован.} \end{cases}$$

Получение функции $f$ является наиболее проблематичной задачей при реализации статистического метода. Ее построение осуществляется на основе теории проверки статистических гипотез.

Для работы с данными, которые имеют двоичный формат, в большинстве случаев проводится манипулирование двумя гипотезами: основной — «блок $b_i$ не был модифицирован», и альтернативной — «блок $b_i$ был модифицирован». При извлечении скрытых данных, функцию $f$ последовательно применяют ко всем блокам контейнера. Если статистика $q(b_i)$ распределения элементов анализируемого блока кон-

тейнера превышает некоторое пороговое значение, то считается, что в блок было встроено «1», в противоположном случае — «0».

Статистические методы сложно применять на практике [3,89]. Причинами этого являются, во-первых, необходимость располагать исчерпывающей статистикой $q(b_i)$ для контейнера-оригинала, на основе которой принимаются решения о возможной модификации исследуемого контейнера (или же его блока); во-вторых, распределение $q(b_i)$ должно быть заранее известно принимающей стороне, а это в большинстве случаев является достаточно сложной задачей.

Питас (I. Pitas) в своей работе [89] предлагает использовать статистический метод для встраивания в полутоновое изображение $C$ размерностью $X \times Y$ цифровой подписи $W$ (ЦВЗ), которая представляет собой псевдослучайный двоичный шаблон размерностью $X \times Y$, в котором количество «единиц» соответствует количеству «нулей»:

$$W = \left\{ w_{x,y},\ x \in \{1,2,...,X\},\ y \in \{1,2,...,Y\} \right\}, \tag{5.51}$$

где $w_{x,y} \in \{0,1\}$.

Оригинальное изображение представляется следующим образом:

$$C = \left\{ c_{x,y},\ x \in \{1,2,...,X\},\ y \in \{1,2,...,Y\} \right\}, \tag{5.52}$$

где $c_{x,y} \in \{0,1,...,255\}$ — уровень интенсивности (яркости) пикселя $(x, y)$.

Множество $C$ разделяется на два подмножества, имеющих равную мощность $P = X \times Y / 2$:

$$A = \left\{ c_{x,y} \in C,\ w_{x,y} = 1 \right\}; \tag{5.53}$$

$$Z = \left\{ c_{x,y} \in C,\ w_{x,y} = 0 \right\}. \tag{5.54}$$

Встраивание ЦВЗ $W$ выполняется путем изменения всех элементов подмножества $A$ на величину положительного целого коэффициента $k > 0$:

$$V = \left\{ c_{x,y} + k,\ c_{x,y} \in A \right\}. \tag{5.55}$$

Изображение $S$ с встроенным ЦВЗ получается путем объединения двух множеств:

$$S = V \cup Z. \tag{5.56}$$

Неизменность зрительного восприятия изображения (незаметность встроенных посторонних данных) предопределяется законом Вебера-Фехнера, а именно тем, что величина коэффициента $k$, прибавляемого к яркости пикселя $c_{x,y} \in A$ для получения множества $V$, в основном является достаточно малой (с учетом $k / c_{x,y} \to 0$) [75].

Автор [89] указывает на возможность довольно точного обнаружения встроенной информации путем исследования изменений, вызванных встраиванием. Главная идея при этом — экспертиза отличий средних значений (математических ожиданий) $\bar{v}$ и $\bar{z}$ двух выделенных подмассивов изображения $V$ и $Z$. К результатам вычисления разности средних значений $\bar{u} = \bar{v} - \bar{z}$ применяется теория проверки гипотезы. Статистика, лежащая в основе критерия:

$$q = \bar{u} / \hat{\sigma}_{\bar{u}}, \tag{5.57}$$

где $\hat{\sigma}^2_{\bar{u}} = \left[ \mathbf{var}(V) + \mathbf{var}(Z) \right] / P$; $\mathbf{var}(\bullet)$ — оценка дисперсии случайных переменных в соответствующем подмножестве.

Основная и альтернативная гипотезы, соответственно, представляют собой:

$\mathbf{H_0}$: ЦВЗ в изображении отсутствует ( $\overline{\boldsymbol{u}} = 0$ );

$\mathbf{H_1}$: в изображение встроен ЦВЗ ( $\overline{\boldsymbol{u}} = \boldsymbol{k}$ ).

Исходя из основной гипотезы, статистика $\boldsymbol{q}$ отвечает распределению Стьюдента с нулевым математическим ожиданием и $(2 \cdot \boldsymbol{P} - 2)$ степенями свободы, которое можно с достаточной точностью аппроксимировать нормальным распределением с нулевым математическим ожиданием и единичной дисперсией.

В случае альтернативной гипотезы, статистика $\boldsymbol{q}$ распределена по так называемому н центрированному распределению Стьюдента с математическим ожиданием $\boldsymbol{k} / \hat{\sigma}_{\overline{u}}$ . При значительном объеме выборки распределение $\boldsymbol{q}$ может быть аппроксимировано нормальным распределением с единичной дисперсией и математическим ожиданием $\boldsymbol{k} / \hat{\sigma}_{\overline{u}}$ .

Во время детектирования ЦВЗ возможны следующие две ошибки:

- *ошибка первого рода:* принято решение о наличии встроенного ЦВЗ, в то время как он в изображении отсутствует («ложная тревога»);
- *ошибка второго рода:* наличие встроенного ЦВЗ не установлено, хотя фактически он в изображении присутствует («пропуск цели»).

Если $\boldsymbol{t}_{1-\alpha}$ является $\boldsymbol{t}$-процентилем, минимизирующим обе ошибки, то[13]

$$\boldsymbol{k} = \left[ 2 \cdot \hat{\sigma}_{\overline{u}} \cdot \boldsymbol{t}_{1-\alpha} \right]. \tag{5.58}$$

Как следствие, перед встраиванием ЦВЗ в контейнер существует возможность задаться степенью достоверности $(1 - \alpha)$, с которой на стадии детектирования можно сделать предположение об отсутствии или наличии встроенного в контейнер ЦВЗ.

Таким образом, предлагается следующий *алгоритм встраивания ЦВЗ:*

1. Подсчитываются значения $\mathbf{var}(\boldsymbol{A})$ и $\mathbf{var}(\boldsymbol{Z})$, которые используются для определения $\hat{\sigma}_{\overline{u}}$, используя формулу $\hat{\sigma}_{\overline{u}}^2 = \left[ \mathbf{var}(\boldsymbol{A}) + \mathbf{var}(\boldsymbol{Z}) \right] / \boldsymbol{P}$.
2. По формуле (5.58) вычисляется значение $\boldsymbol{k}$. Следует заметить, что использованное в данной формуле квантование изменяет степень достоверности, сводя ее к некоторому значению $(1 - \alpha')$. Кроме того, авторами было сделано предположение, что $\mathbf{var}(\boldsymbol{V}) = \mathbf{var}(\boldsymbol{A})$, что не является полностью справедливым по причине отсечений, возникающих в случае, когда результат действия $\boldsymbol{c}_{x,y} + \boldsymbol{k}$ выходит за пределы разрешенного диапазона $[0; 255]$.
3. Создается «подписанное» ЦВЗ изображение $\boldsymbol{S}$ путем замены подмножества $\boldsymbol{A}$ из множества $\boldsymbol{C}$ на подмножество $\boldsymbol{V}$ (формулы (5.55), (5.56)).

*Алгоритм детектирования ЦВЗ* выглядит следующим образом:

1. Определяются математические ожидания $\overline{\boldsymbol{v}}$ и $\overline{\boldsymbol{z}}$ выделенных подмассивов $\boldsymbol{V}$ и $\boldsymbol{Z}$, по которым вычисляется разница $\overline{\boldsymbol{u}} = \overline{\boldsymbol{v}} - \overline{\boldsymbol{z}}$ .
2. Определяются оценки дисперсии $\mathbf{var}(\boldsymbol{V})$ и $\mathbf{var}(\boldsymbol{Z})$, на основе которых проводится расчет параметра $\hat{\sigma}_{\overline{u}}$ (см. комментарий к формуле (5.57)).

---

[13] Процентиль является разновидностью квантиля порядка $\rho$ одномерного распределения — такого значения $\boldsymbol{t}_\rho$ случайной величины $\boldsymbol{t}$, для которого $\mathrm{P}\{\boldsymbol{t} < \boldsymbol{t}_\rho\} = \boldsymbol{F}(\boldsymbol{t}_\rho) = \rho$, $(0 < \rho < 1)$. Процентили $\boldsymbol{t}_{0.01}$, $\boldsymbol{t}_{0.02}$, ..., $\boldsymbol{t}_{0.99}$ делят область изменения $\rho$ на 100 интервалов, попадания в которые имеют равные вероятности. Более подробно см., например, в [105].

3. На основании (5.57) создается статистика $q$, которая сравнивается с процентилями $t_{1-\alpha}$. В случае, если $q < t_{1-\alpha}$, делают вывод про отсутствие ЦВЗ в изображении. В противном случае с вероятностью $(1 - \alpha)$ ЦВЗ в изображении присутствует.

Рассмотрим возможный пример реализации данного метода в программе MathCAD.

**Шаг 1**

Пусть изображение-контейнер представляет собой графический файл C.bmp (см. рис. 5.35, **C**): **C** := **READBMP**("C.bmp"). При этом **X** := **rows**(**C**), **X** = 128; **Y** := **cols**(**C**), **Y** = 128.

Черно-белое (двоичное) изображение ЦВЗ представлено на рис. 5.57: **W** := **READBMP**("W.bmp"); **rows**(**W**) = 128; **cols**(**W**) = 128.

Количество нулевых и значащих элементов ЦВЗ одинаковое, исходя из чего среднее значение множества **W**: **mean**(**W**) = 0,5.


**Рис. 5.57. ЦВЗ, подлежащий внедрению в контейнер**

**Шаг 2**

С помощью программных модулей (М.98*а*,*б*) делим множество **C** на два подмножества одинаковой мощности **P** := **X**·**Y**/2 = 8192 в соответствии с формулами (5.53) и (5.54):

```
A := | for x ∈ 1..X
     |   for y ∈ 1..Y
     |     A[x,y] ← C[x,y]  if W[x,y] = 1
     |     A[x,y] ← 0       otherwise
     | A
```

(М.98*а*)

```
Z := | for x ∈ 1..X
     |   for y ∈ 1..Y
     |     Z[x,y] ← C[x,y]  if W[x,y] = 0
     |     Z[x,y] ← 0       otherwise
     | Z
```

(М.98*б*)

Графическая интерпретация результата разделения изображена на рис. 5.58.

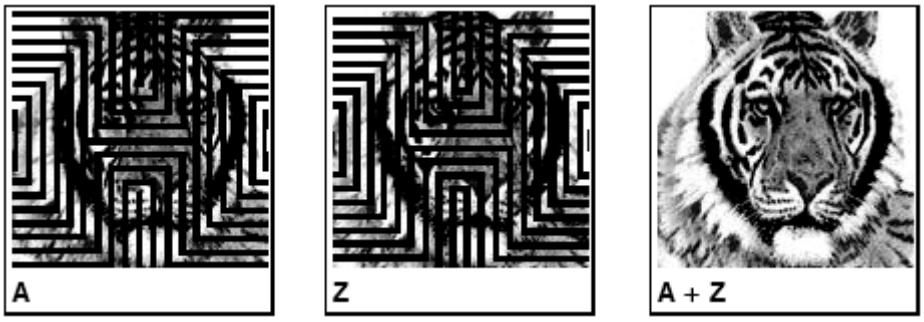



**Рис. 5.58. Графическая интерпретация подмножеств A и Z, а также их объединения A+Z**

**Шаг 3**

Используя программные модули (М.99*а*,*б*) и (М.100*а*,*б*), определяем, соответственно, средние значения (**mean**) и дисперсии (**var**) подмножеств **A** и **Z**.

$$\mathbf{A_{mean}} := \left| \begin{array}{l} \Sigma A \leftarrow 0 \\ \textbf{for}\ \ x \in 1..X \\ \quad \textbf{for}\ \ y \in 1..Y \\ \qquad \Sigma A \leftarrow \Sigma A + A_{x,y} \ \ \textbf{if}\ \ W_{x,y} = 1 \\ \dfrac{\Sigma A}{P} \end{array} \right. \qquad (М.99\,а)$$

$$\mathbf{Z_{mean}} := \left| \begin{array}{l} \Sigma Z \leftarrow 0 \\ \textbf{for}\ \ x \in 1..X \\ \quad \textbf{for}\ \ y \in 1..Y \\ \qquad \Sigma Z \leftarrow \Sigma Z + Z_{x,y} \ \ \textbf{if}\ \ W_{x,y} = 0 \\ \dfrac{\Sigma Z}{P} \end{array} \right. \qquad (М.99\,а)$$

$$\mathbf{A_{var}} := \left| \begin{array}{l} \Sigma A' \leftarrow 0 \\ \textbf{for}\ \ x \in 1..X \\ \quad \textbf{for}\ \ y \in 1..Y \\ \qquad \Sigma A' \leftarrow \Sigma A' + \left(A_{x,y} - A_{mean}\right)^2 \ \ \textbf{if}\ \ W_{x,y} = 1 \\ \dfrac{\Sigma A'}{P} \end{array} \right. \qquad (М.100\,а)$$

$$\mathbf{Z_{var}} := \left| \begin{array}{l} \Sigma Z' \leftarrow 0 \\ \textbf{for}\ \ x \in 1..X \\ \quad \textbf{for}\ \ y \in 1..Y \\ \qquad \Sigma Z' \leftarrow \Sigma Z' + \left(Z_{x,y} - Z_{mean}\right)^2 \ \ \textbf{if}\ \ W_{x,y} = 0 \\ \dfrac{\Sigma Z'}{P} \end{array} \right. \qquad (М.100\,б)$$

В результате имеем:

$\mathbf{A_{mean}} = 146{,}6$; $\mathbf{Z_{mean}} = 146{,}4$; $\mathbf{A_{var}} = 7398{,}8$; $\mathbf{Z_{var}} = 7299{,}3$.

По полученным данным проводим расчет параметра $\hat{\sigma}_{\bar{u}}$:

$$\sigma\text{^}\mathbf{u} := \sqrt{\frac{\mathbf{A_{var}} + \mathbf{Z_{var}}}{\mathbf{P}}}; \qquad \sigma\text{^}\mathbf{u} = 1{,}339.$$

**Шаг 4**

В соответствии с формулой (5.58) проводим вычисление значения **k**. Процентиль $\mathbf{t}_{1-\alpha}$ получим исходя из того, что количество степеней свободы

$$\mathbf{d} := 2 \cdot \mathbf{P} - 2 = 16382 \approx \infty.$$

При этом

$$\mathbf{t_{1-\alpha}}(\alpha) := \mathbf{qt}\left(1 - \frac{\alpha}{2}, \mathbf{d}\right),$$

где **qt**(•) — встроенная функция MathCAD расчета обратного кумулятивного распределения вероятностей (квантиля) для распределения Стьюдента.

График зависимости $\mathbf{t}_{1-\alpha}(\alpha)$ приведен на рис. 5.59.

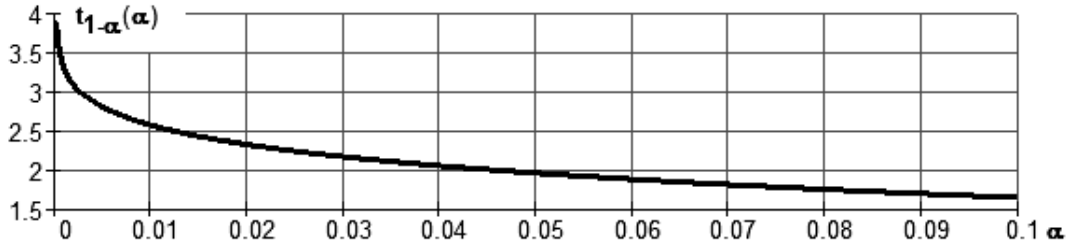



**Рис. 5.59. Зависимость процентиля $t_{1-\alpha}$ от риска в допущении ошибки α**

Задавшись значением $\alpha := 0.001$, имеем $\mathbf{t}_{1-\alpha}(\alpha) = 3.291$, в результате чего

$$\mathbf{k}(\alpha) := \mathbf{ceil}\left(\left(2\cdot\sigma\mathbf{\hat{}u}\cdot\mathbf{t_{1-\alpha}}(\alpha)\right)\right), \qquad \mathbf{k}(\alpha) = 9.$$

**Шаг 5**

В соответствии с (5.55) проводим встраивание ЦВЗ в подмножество **A**, изменяя элементы последнего на величину коэффициента **k** — программный модуль (М.101).

```
V := | k ← k(α)
     | for x ∈ 1..X
     |   for y ∈ 1..Y
     |     if W[x,y] = 1
     |       V[x,y] ← A[x,y] + k
     |       V[x,y] ← 255  if V[x,y] > 255
     |     V[x,y] ← 0  otherwise
     | V
```

(М.101)

Использование в данном модуле оператора проверки условия $\mathbf{V}_{x,y} > 255$ позволяет предусмотреть возможность выхода значения элемента $\mathbf{V}_{x,y}$ из диапазона [0, 255]. Вид полученного подмножества **V**, а также «подписанное» изображение **S** := **V** + **Z** приведены на рис. 5.60.

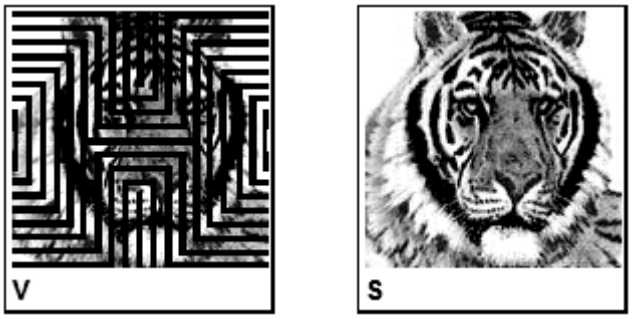



**Рис. 5.60. Графическая интерпретация подмножеств V и заполненного контейнера S**

**Шаг 6**

Для детектирования ЦВЗ в контейнере должны быть известны:

- изображение **S***, подозреваемое на наличие встроенного ЦВЗ, и его размерность **X***×**Y***;
- изображение **W***, которое, как предполагается, использовалось в качестве ЦВЗ.

Используя программные модули, аналогичные модулям (М.98*а*,*б*), делим множество **S*** на подмножества **V*** и **Z*** одинаковой мощности **P*** := **X***·**Y***/2 = 8192, исходя из соответствующих значений элементов множества **W***.

Используя программные модули, подобные модулям (М.99) и (М.100), рассчитываем средние значения (**mean**) и дисперсии (**var**) полученных подмножеств:

$$\mathbf{V^*_{mean}} = 154{,}4;\quad \mathbf{Z^*_{mean}} = 146{,}4;\qquad \mathbf{V^*_{var}} = 7149{,}1;\quad \mathbf{Z_{var}} = 7299{,}3.$$

По полученным данным находим значение параметров $\overline{u}$ и $\hat{\sigma}_{\overline{u}}$ (см. формулу (5.57)):

$$\mathbf{U_{mean}} := \mathbf{V^*_{mean}} - \mathbf{Z^*_{mean}},\qquad \mathbf{U_{mean}} = 8.0;$$

$$\sigma^*\hat{}_{\mathbf{u}} := \sqrt{\frac{\mathbf{V^*_{var}} + \mathbf{Z^*_{var}}}{\mathbf{P^*}}},\qquad \sigma^*\hat{}_{\mathbf{u}} = 1{,}328.$$

Таким образом, при $\alpha^* := 0.001$, имеем:

$$\mathbf{q} := \frac{\mathbf{U_{mean}}}{\sigma^*\hat{}_{\mathbf{u}}},\qquad \mathbf{q} = 6{,}025 \;>\; \mathbf{t^*_{1-\alpha}}(\alpha^*) = 3{,}291.$$

Из чего можно сделать вывод о том, что с вероятностью 99,9% в изображении **S*** присутствует ЦВЗ **W***.

### 5.3.5.2. Структурные методы

Методы, получившие на сегодняшний день наибольшее распространение, в основном используют информационный избыток на уровне пикселей (пространственную область) или же осуществляют преобразование в частотной области изображения. Ю.М. Коростиль и М.Е.Шелест предложили метод, в котором скрытие информации выполняется на содержательном уровне с использованием структурных и информационных параметров изображения [3, 91]. Предложенный метод является развитием известной стеганографической технологии — *семаграмм*. Семаграммы представляют собой сообщения, в которых шифробозначениями являются любые символы, кроме букв и цифр. Эти сообщения могут быть переданы, например, в рисунке, содержащем точки и тире для чтения по коду Морзе.

Суть же метода состоит в проведении последовательных преобразований отдельных фрагментов графического изображения, что в конечном итоге приводит к формированию скрываемого текста.

В структурных методах выделяют следующие этапы стеганопреобразования:

1. Преобразование защищаемого секретного сообщения $\boldsymbol{M}$ в цифровую форму $\boldsymbol{D_M}$ (например, с помощью любого криптографического кодирования), что подразумевает под собой шифрование текста со всеми соответствующими атрибутами.
2. Преобразование последовательности чисел $\boldsymbol{D_M}$ в графическую структуру $\boldsymbol{G_M}$ (граф, пиктограмму и т.п.), которая тем или иным способом может быть подвергнута формальному описанию.

3. Преобразование графической структуры $G_M$ в визуальную информационную среду $V_M$ (например, мультимедийную или же программную).
4. Использование совокупности методов и соответствующих процедур, с помощью которых формируется сюжет из визуальных образов со встроенными в них скрытыми сообщениями.

Следовательно, всю последовательность преобразований можно записать следующим образом:

$$M \Rightarrow D_M \Rightarrow G_M \Rightarrow V_M \Rightarrow S_M,$$

где $S_M$ — описание сюжета, который состоит из отдельных графических образов.

## 5.4. Скрытие данных в аудиосигналах

Особое развитие получили цифровые методы стеганографии в аудиосреде. Скрытие данных в звуковых (аудио-) сигналах является особенно перспективным, поскольку слуховая система человека (ССЧ), работает в сверхшироком динамическом диапазоне. ССЧ воспринимает более чем миллиард к одному в диапазоне мощности и более чем тысяча к одному в частотном диапазоне [14]. Кроме этого, высокой является и чувствительность к аддитивному флуктуационному (белому) шуму. Отклонения в звуковом файле могут быть выявлены вплоть до одной десятимиллионной (на 70 дБ ниже уровня внешних шумов).

Несмотря на это, существуют определенные возможности для скрытия информации и в аудиосреде. Хотя ССЧ и имеет широкий динамический диапазон, она характеризуется достаточно малым разностным диапазоном. Как следствие, громкие звуки содействуют маскировке тихих звуков. Кроме того, ССЧ не способна различать абсолютную фазу, распознавая только относительную. Наконец, существуют некоторые виды искажений, вызванных окружающей средой, которые настолько обычны для слушателя, что в большинстве случаев им игнорируются.

Подобные особенности слухового аппарата человека позволяют удачно использовать аудиосреду с целью стеганографической защиты конфиденциальной информации. Значительный вклад в развитие аудиостеганографии сделали Бендер (W. Bender), Моримото (N. Morimoto) и др. Поэтому при дальнейшем изложении предлагается рассмотреть основные сведения и методы, изложенные данными авторами в своей работе [14].

### *5.4.1. Кодирование наименее значащих бит (временная область)*

Кодирование младших разрядов является простейшим способом внедрить конфиденциальные данные в иные структуры данных. Используя звуковой сигнал, путем замены НЗБ каждой точки осуществления выборки, представленной двоичной последовательностью, можно зашифровать значительный объем информации.

Теоретически, пропускная способность стеганоканала составляет 1 Кб/сек на 1 кГц в канале без помех, битовая скорость передачи данных составит 8 Кб/сек в последовательности, которая оцифрована с частотой 8 кГц, и 44 Кб/сек в последовательности с частотой дискретизации 44 кГц. Платой за высокую пропускную способность канала является ощутимый на слух низкочастотный шум. Слышимость данного шума непосредственно зависит от содержимого сигнала-контейнера. Например, шум зрителей во время эфира спортивного соревнования в достаточной

степени маскировал бы шум наименьших бит, модифицированных кодированием. Однако указанный шум будет ощутимым на слух при использовании в качестве контейнера аудиозаписи игры струнного квартета. Для компенсации этого изменения целесообразным будет использование адаптивной аттенюации данных.

Главный недостаток метода кодирования НЗБ, как и в случае с графическим контейнером, — это его слабая стойкость к посторонним воздействиям. Встроенная информация может быть разрушена из-за наличия шумов в канале, в результате передискретизации выборки и т.п., за исключением случаев, когда информация встраивалась с внесением избыточности. Однако последнее, обеспечивая приемлемую стойкость к помехам, приводит к уменьшению скорости передачи данных, зачастую на один/два порядка. На практике метод полезен только в замкнутых, полностью цифровых средах, не требующих дополнительного преобразования.

Как и в случае с использованием в качестве контейнера файла изображения, перед импортом аудиоконтейнера в документ MathCAD его необходимо подготовить в соответствующем звуковом (музыкальном) редакторе. Необходимо отметить, что программа MathCAD поддерживает только файлы WAV-формата импульсно-кодово-модулированных (ИКМ) сигналов (pulse-code modulated, PCM-signals), который, однако, является одним из самых распространенных на сегодняшний день.

Для предотвращения возможности сравнения нарушителем перехваченного контейнера с аудиофайлами, имеющимися в его распоряжении, а на этом основании — доказывания факта существования скрытого сообщения и, возможно, извлечения или модификации последнего, в качестве контейнера рекомендуется использовать именно уникальные (созданные собственноручно) записи.

Подготовленный аудиоконтейнер следует поместить в текущую для создаваемого документа MathCAD директорию. В нашем случае предварительно был создан файл ИКМ-сигнала "C.wav".

**Шаг 1**

Для получения информации относительно WAV-файла используется встроенная функция MathCAD **GETWAVINFO**("файл"), где под аргументом "файл" подразумевается текстовая строка, содержащая в себе имя файла (или же полный путь и имя файла), который предусматривается использовать в качестве контейнера.

Указанная функция возвращает четырехэлементный вектор с информацией о файле, который выступил ее аргументом. Первый элемент вектора характеризует *количество каналов*; второй — *частоту дискретизации* (в герцах); третий — *количество бит, которыми кодируется один отсчет* (определяющее количество уровней квантования); четвертый — *среднее количество бит в секунду,* которое должен обрабатывать аудиопроигрыватель, чтобы воспроизводить этот файл в реальном времени.

Ниже приведем два возможных варианта использования данной функции:

$$\mathbf{GETWAVINFO}(\text{"C.wav"}) = \begin{pmatrix} 2 \\ 22050 \\ 16 \\ 88200 \end{pmatrix}; \quad \text{или} \quad \begin{pmatrix} N_К \\ f_Д \\ Q \\ B \end{pmatrix} \equiv \mathbf{GETWAVINFO}(\text{"C.wav"}); \quad \begin{aligned} N_К &= 2 \text{ канала;} \\ f_Д &= 22050 \text{ Гц;} \\ Q &= 16 \text{ бит;} \\ B &= 88200 \text{ бит/с.} \end{aligned}$$

**Шаг 2**

Пользуясь полученной информацией, найдем временной вектор, отвечающий амплитудам аудиосигнала в отдельные отсчеты дискретизации. Данные амплитуды могут быть считаны с помощью функции **READWAV**("файл"). Данная функция возвращает массив, каждый столбец которого представляет собой отдельный канал (так, например, для моносигнала массив будет содержать только 1 столбец, для стерео — 2 и т.д.), а каждая строка массива отвечает моменту времени, который определяется номером отсчета и частотой дискретизации сигнала.

Отдельный элемент массива, в зависимости от количества бит кодирования **Q**, может принимать значения или от 0 до $2^8 - 1 = 255$ (при **Q** = 1...8), или же от $-2^{16-1} = -32768$ до $2^{16-1} - 1 = 32767$ (при **Q** = 9...16).

Пусть **C** := **READWAV**("C.wav"). Фрагмент импортированного звука (коды отсчетов от 1000-го по 1010-й в десятичном и двоичном виде) изображен на рис. 5.61.

| i = | $C_{i,1}$ = | $C_{i,2}$ = | $C_{i,1}$ = | $C_{i,2}$ = |
|---|---|---|---|---|
| 1000 | 5055 | 6154 | 1001110111111b | 1100000001010b |
| 1001 | 5213 | 6071 | 1010001011101b | 1011110110111b |
| 1002 | 4947 | 6125 | 1001101010011b | 1011111101101b |
| 1003 | 4131 | 5371 | 1000000100011b | 1010011111011b |
| 1004 | 3131 | 4088 | 110000111011b | 111111111000b |
| 1005 | 2446 | 3274 | 100110001110b | 110011001010b |
| 1006 | 1842 | 2485 | 11100110010b | 100110110101b |
| 1007 | 1174 | 1631 | 10010010110b | 11001011111b |
| 1008 | 304 | 494 | 100110000b | 111101110b |
| 1009 | -953 | -1127 | -1110111001b | -10001100111b |
| 1010 | -2466 | -3080 | -100110100010b | -110000001000b |

**Рис. 5.61. Фрагмент импортированного аудиофайла в виде массива квантованных амплитуд**

Общее количество отсчетов на каждый из каналов: $\aleph_{\text{С/кан.}}$ := **rows**(**C**), где **rows**(**C**) — функция, возвращающая количество строк массива **C**. В нашем случае, например, $\aleph_{\text{С/кан.}} = 20191$ отсчет на канал.

Определим временны*е* координаты каждого из отсчетов: **n** := 1 .. $\aleph_{\text{С/кан.}}$, $t_n := n / f_Д$ (сек.). Значения временны*х* интервалов, которые отвечают приведенному выше (рис. 5.61) фрагменту отсчетов импортированного звука, приведены на рис. 5.62.

Интервал дискретизации

$$\Delta t := f_Д^{-1} = t_1 - 0 = t_2 - t_1 = \ldots = t_n - t_{n-1}.$$

Следовательно, $\Delta t = 45.351 \cdot 10^{-6}$ сек. Общая продолжительность звучания аудиофайла:

$$t_\Sigma := \max(t) = 915.692 \cdot 10^{-3} \text{ сек.}$$

Очевидно, что разделив общее количество отсчетов $\aleph_{\text{С/кан.}}$ на общую продолжительность звучания $t_\Sigma$, мы получим значение частоты дискретизации $f_Д$.

| $t_i$ = |
|---|
| $45.351 \cdot 10^{-3}$ |
| $45.397 \cdot 10^{-3}$ |
| $45.442 \cdot 10^{-3}$ |
| $45.488 \cdot 10^{-3}$ |
| $45.533 \cdot 10^{-3}$ |
| $45.578 \cdot 10^{-3}$ |
| $45.624 \cdot 10^{-3}$ |
| $45.669 \cdot 10^{-3}$ |
| $45.714 \cdot 10^{-3}$ |
| $45.760 \cdot 10^{-3}$ |
| $45.805 \cdot 10^{-3}$ |

**Рис. 5.62. Фрагмент отсчетов дискретизации импортированного аудиофайла**

Владея информацией о временны*х* координатах каждого из отсчетов и соответствующие этим отсчетам амплитуды (в квантованном виде) можно изобразить «осциллограмму» импортированного аудиофайла (рис. 5.63).

**Шаг 3**

В качестве скрываемого сообщения используем следующее:

**M** := **str2vec**("© Пузыренко А.Ю., 2005 г.").

Аналогично к тому, как это было показано выше (программные модули (М.1) и (М.2)), проведем криптографическое кодирование сообщения, результатом чего будет массив **M_cod**.

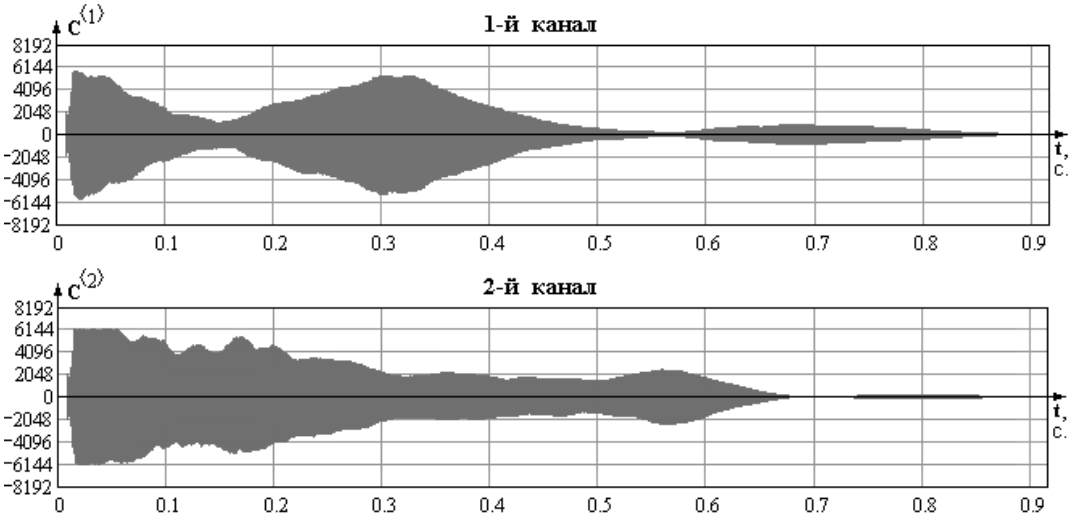



**Рис. 5.63. Временны*е* диаграммы каналов ИКМ сигнала "C.wav"**

**Шаг 4**

Для скрытия одного символа сообщения достаточно 8 бит звука. Для того чтобы при извлечении скрытых данных в полученной строке символов можно было четко определить начало и конец полезного сообщения, введем соответствующие метки, которые бы тем или иным образом указывали на границы именно полезного содержания. Примем, что стартовая метка $\mu_s$ определяет порядковый номер элемента контейнера, начиная с которого в последний будут встраиваться данные. Пусть $\mu_s$ := 74. Метка $\mu_e$ будет сигнализировать о завершении смысловой части среди распакованных символов, $\mu_e$ := "KiHeu,6".

Ограничивающую метку $\mu_e$, предварительно преобразовав ее в вектор ASCII-кодов, введем в текст закодированного сообщения:

**Me** := **stack**(**M_cod**, **str2vec**($\mu_e$)).

Таким образом, общее количество символов в сообщении, которое подлежит скрытию: **rows**(**Me**) = 32 симв. Количество необходимых для этого бит (8 бит на символ): 8·**rows**(**Me**) = 256 бит.

Общее количество НЗБ аудиоконтейнера: **rows**(**C**)·**cols**(**C**) = 40382 >> 256 бит. Следовательно, аудиофайл имеет достаточный объем и пригоден для использования в качестве контейнера для данного сообщения.

**Шаг 5**

Для дальнейших расчетов также будет необходимо преобразование форматов чисел из десятичного в двоичный и назад. Для этого используем программные модули (М.102) и (М.103), которые подобны модулям (М.3) и (М.4) с той только разницей, что верхняя граница индексирования в последних (восьмерка) заменяется на **rows**(**x**) и **Q** соответственно. Кроме этого, при преобразовании формата аргумента учитывается его знак.

Отметим, что в модуле (М.103) знаковая функция **sign**(**x**) возвращает 0 (если **x** = 0); 1 (если **x** > 0); –1 (если **x** < 0). Это позволяет сохранить знак числа **x** при переведении его формата в двоичный.

$$\mathbf{B2D}(\mathbf{x}) := \sum_{\mathbf{i}=1}^{\mathbf{rows}(\mathbf{x})} \left(\mathbf{x}_{\mathbf{i}} \cdot 2^{\mathbf{i}-1}\right) \tag{М.102}$$

$$\begin{aligned}
\mathbf{D2B}(\mathbf{x}) := \;& \mathbf{знак} \leftarrow \mathbf{sign}(\mathbf{x}) \\
& \textbf{for}\ \ \mathbf{i} \in 1..\mathbf{Q} \\
& \quad \mathbf{V}_{\mathbf{i}} \leftarrow \mathbf{mod}(|\mathbf{x}|, 2) \\
& \quad \mathbf{x} \leftarrow \mathbf{floor}\left(\frac{|\mathbf{x}|}{2}\right) \\
& \mathbf{знак} \cdot \mathbf{V}
\end{aligned} \tag{М.103}$$

Внесение бит сообщения в контейнер будем проводить с переменным шагом, величина которого будет определяться количеством единиц в двоичном значении номера элемента контейнера, который модифицировался предварительно. Для определения величины шага используем программный модуль (М.15).

**Шаг 6**

Для удобства дальнейших действий, проведем векторизацию массива **C** (в случае, если количество каналов в звуке больше одного) — программный модуль (М.104), предварительно выполнив перестановку элементов каждого из столбцов в обратном порядке (с помощью функции **reverse**(•)), что повысит степень скрытости встроенной информации.

$$\begin{aligned}
\mathbf{Cv} := \;& \mathbf{Cv} \leftarrow \mathbf{reverse}\left(\mathbf{C}^{\langle 1 \rangle}\right) \\
& \textbf{for}\ \ \mathbf{i} \in 2..\mathbf{N_K} \qquad \textbf{if}\ \ \mathbf{N_K} \geq 2 \\
& \quad \mathbf{Cv} \leftarrow \mathbf{stack}\left(\mathbf{Cv}, \mathbf{reverse}\left(\mathbf{C}^{\langle \mathbf{i} \rangle}\right)\right) \\
& \mathbf{Cv}
\end{aligned} \tag{М.104}$$

На основе массива **Cv** формируем новый массив, который уже будет содержать скрытое закодированное сообщение. Для этого применим программный модуль (М.105). В модуле (М.105) каждый символ кодированного сообщения (операция цикла **for** $\tau \in 1..$**rows**(**Me**)) переводится в двоичный формат (переменная **b**), каждый разряд которого записывается вместо НЗБ числа, отвечающего значению амплитуды того или иного отсчета. При этом элементы массива **Cv** проходятся не последовательно, а с переменным шагом, величина которого в данном случае обуславливается количеством единиц в двоичном представлении номера элемента, который модифицировался предварительно.

```
Sv := | Sv ← Cv
      | Z ← μ_S
      | for τ ∈ 1.. rows(Me)
      |   | b ← D2B(Me_τ)
      |   | for i ∈ 1.. 8
      |   |   | Z ← Z + step(D2B(Z))
      |   |   | знак ← 1   if  Cv_Z ≥ 0
      |   |   | знак ← −1  if  Cv_Z < 0
      |   |   | P ← D2B(Cv_Z)
      |   | * | P_1 ← знак·b_i
      |   |   | Sv_Z ← B2D(P)
      | Sv
```

(M.105)

Функция **step**(•) модуля (M.15) вычислялась при значении $\kappa = 20$ (в этом случае должно учитываться общее количество НЗБ, которое необходимо для встраивания сообщения, и имеющееся количество элементов массива контейнера). Стартовый элемент встраивания задается меткой $\mu_S$.

Значение элемента массива **Cv**, в который будет вестись запись, предварительно переводится в двоичный формат (переменная **P**)). Также учитывается знак амплитуды отсчета. После проведенного изменения НЗБ модифицированное двоичное число **P** переводится в формат десятичного и записывается в соответствующую позицию вектора **Sv**, который в начале модуля был принят равным вектору **Cv**.

**Шаг 7**

Проводим сворачивание вектора **Sv** в массив **S**, имеющий размерность первичного массива **C**. Одновременно выполняем обратную перестановку, возвращая элементы отсчетов на свои места:

```
S := for i ∈ 1.. cols(C)
       S^⟨i⟩ ← reverse[submatrix[Sv, (i − 1)·rows(C) + 1, i·rows(C), 1, 1]]
```

(M.106)

На рис. 5.64 изображены восстановленные по соответствующим значениям амплитуд отсчетов временны*е* диаграммы сигнала аудиоконтейнера, который содержит скрытое сообщение.

Сравнивая рис. 5.64 и рис. 5.63 можно сделать вывод об их полной визуальной идентичности, а это дает определенные основания утверждать об идентичности заполненного и исходного контейнеров на слух.

Для того чтобы убедиться в этом, необходимо записать массив **S** в файл, что можно сделать встроенной функцией **WRITEWAV**("файл", частота дискретизации, количество бит квантования):

**WRITEWAV**("S_LSB.wav", $f_д$, **Q**) := **S**.

**Шаг 8**

Чтобы исследовать влияние на степень скрытости того факта, в какой из разрядов числа, характеризующего уровень сигнала определенного отсчета, будет вноситься конфиденциальная информация, в модуле (M.105) в помеченной символом

"*" строке вместо индекса "1" следует внести индекс, отвечающий разряду, который предполагается модифицировать.

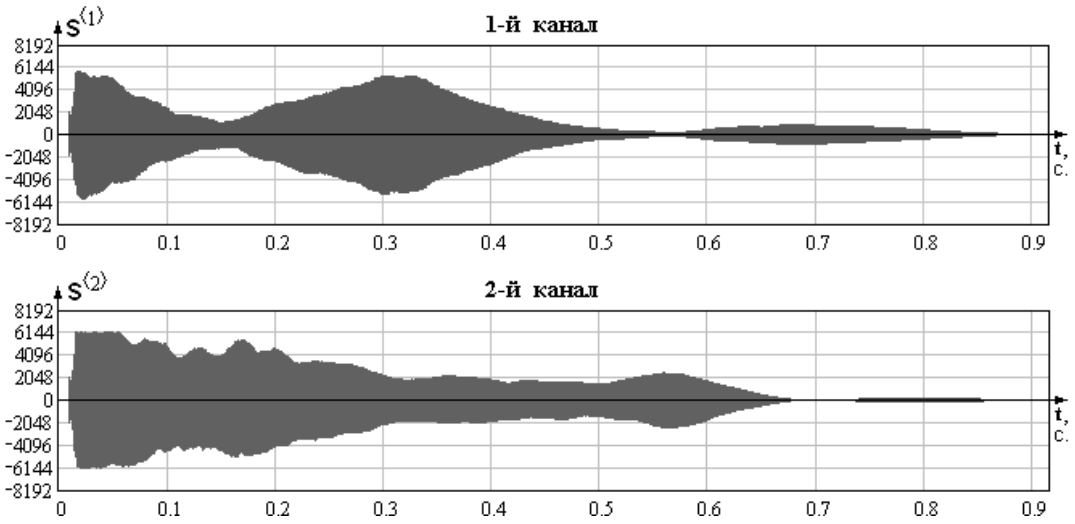



**Рис. 5.64. Временны*е* диаграммы каналов ИКМ сигнала с модифицированными НЗБ**

Среднестатистический человек не будет ощущать на слух разницы между звучанием оригинала и контейнера с сообщением, если в качестве «носителей» использовать не только самый младший, но и предыдущие несколько бит (особенно, если используется 16-битовое квантование и в аудиосигнале отсутствуют значительные участки пауз или участки с длительным постоянным уровнем звучания). Следовательно, еще одним из возможных вариантов защиты можно назвать использование изменяемой по определенному закону записи в эти несколько бит. Или же, снижая уровень скрытости, можно в несколько раз увеличить пропускную способность создаваемого аудиостеганоканала. На рис. 5.65 изображен результат внесения данных в 13-й бит числа квантованного отсчета.

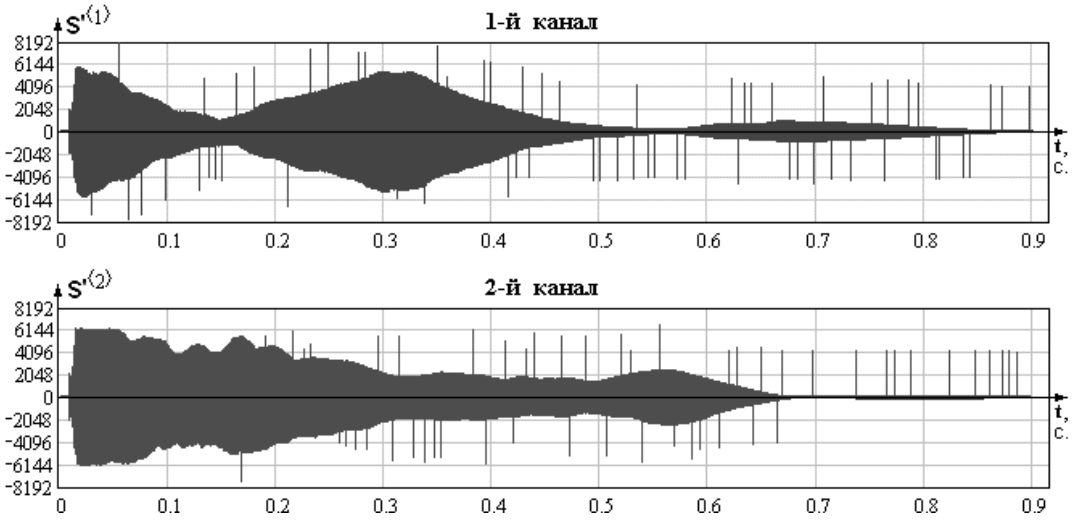



**Рис. 5.65. Временны*е* диаграммы каналов ИКМ сигнала при внесении скрываемых данных в 13-й бит числа, характеризующего уровень сигнала**

**Шаг 9**

Рассмотрим процесс извлечения скрытого сообщения.

Для этого формируем массив значений амплитуд дискретных отсчетов, а также считываем служебную информацию из аудиоконтейнера. Полученный массив **S*** разворачиваем в вектор **Sv***, изменяя порядок элементов в столбцах (используется программный модуль (М.104) с соответствующими заменами **Cv** → **Sv***, **C** → **S***).

**S*** := **READWAV**("S_LSB.wav");

$$\begin{pmatrix} N_к \\ f_д \\ Q \\ B \end{pmatrix} \equiv \mathbf{GETWAVINFO}(\text{"S\_LSB.wav"}) \qquad \begin{aligned} N_к &= 2 \text{ канала;} \\ f_д &= 22050 \text{ Гц;} \\ Q &= 16 \text{ бит;} \\ B &= 88200 \text{ бит/сек.} \end{aligned}$$

Используя программный модуль (М.107), проводим извлечение скрытого кодированного сообщения.

```
Mf* := | Z ← μ*_S
       | err ← 0
       | for τ ∈ 1.. rows(Sv*)
       |    for i ∈ 1.. 8
       |       Z ← Z + step(D2B(Z))
       |       if Z ≥ rows(Sv*)
       |          err ← 1
       |          break
       |       P ← D2B(Sv*_Z)
       |       b_i ← |P_1|
       |    Mf*_τ ← B2D(b)
       |    Mf*_τ ← Mf*_τ + 32.5 if Mf*_τ < 32
       |    break if err = 1
       | Mf*
```

(М.107)

Поскольку в подавляющем большинстве случаев получателю заранее не известно, сообщение какой длины было скрыто в контейнере, принимаются во внимание все элементы вектора **Sv***.

Значения каждого элемента формируемого при этом вектора **Mf*** представляют собой коды символов предположительного сообщения, которые вычисляются в обратном к (М.105) порядке: модуль каждого младшего разряда восьмерки обращенных в двоичный формат элементов вектора **Sv***, выбранных с учетом переменного шага и известного значения метки $\mu^*_S := 74$, формирует двоичное число кода символа, формат которого в дальнейшем преобразуется в десятичный. Полученное число присваивается $\tau$-му элементу вектора **Mf***.

По причине вышеуказанной невозможности обработки MathCAD 12-ой версии символов, ASCII-коды которых имеют значения [0...9, 11, 12, 14...31], дополнительно вносится изменение значений 0, 1, 2, ..., 31 путем добавления к каждому из них

коэффициента 32,5. Причина выбора именно такого коэффициента объяснена в комментариях к программным модулям (М.10) и (М.11).

Переменная **err** необходима для прерывания цикла приращения **i** в случае выхода аргумента **Z** за пределы массива **Sv***. Номера элементов вектора **Mf***, значения которых являются дробными, формируют массив **N** (см. модуль (М.11)).

Владея информацией о том, что из себя представляет конечная метка полезного сообщения (в нашем случае $\mu^*_e$ = "KiHeu,6"), выделяем его с извлеченного сообщения, используя программный модуль (М.108).

```
M_cod* := | e ← 0
          | β_e ← strlen(μ*_e)
          | Mf* ← vec2str(Mf*)
          | for τ ∈ 1 .. strlen(Mf*)
          |   | e ← τ  if substr(Mf*, τ, β_e) = μ*_e ∧ e = 0
          |   | break  if e ≠ 0
          | Mf* ← substr(Mf*, 0, e)
          | M_cod* ← str2vec(Mf*)
          | for n ∈ 1 .. cols(N)
          |   for i ∈ 1 .. rows(N)
          |     | break  if N_i,n = 0
          |     | M_cod*_N_i,n ← n − 1  if 0 < N_i,n ≤ rows(M_cod*)
          | M_cod*
```
(М.108)

Вектор ASCII-кодов **Mf***, предварительно преобразованный функцией **vec2str**(•) в соответствующую символьную строку, последовательно проходится в поиске конечной метки $\mu^*_e$. Поиск осуществляется путем сравнения значения метки с выделенной частью строки символов аналогично тому, как это было реализовано в (М.12). Если совпадение произошло, параметру **e** присваивается значение индекса символа, завершающего содержательную часть сообщения ($\tau$), а цикл поиска прерывается. Извлечение части строки **Mf***, ограниченной символами с индексами 0 и **e**, проводится **с** помощью функции **substr**(•).

После преобразования выделенной символьной строки в вектор кодов выполняется восстановление элементов, значения которых в модуле (М.107) были принудительно изменены на дробные.

**Шаг 10**

Распакованное сообщение декодируем, применяя программные модули (М.13) и (М.14). Декодированное сообщение записываем в файл:

**WRITEBIN**("M_dec.txt","byte",0) := **M***.

Результаты вычисления показателей визуального искажения, приведенных в главе 3, сведены в табл. 5.6.

### *5.4.2. Метод фазового кодирования (частотная область)*

Основная идея метода фазового кодирования состоит в замене фазы исходного звукового сегмента на опорную фазу, характер изменения которой отражает собой

данные, которые необходимо скрыть. Для того чтобы сохранить разностную фазу между сегментами, фазы последних соответствующим образом согласовываются.

Фазовое кодирование, когда оно может быть использовано, является одним из наиболее эффективных методов по критерию отношения сигнал/воспринимаемый шум. Существенное изменение соотношения фаз между каждыми частотными составляющими приводит к значительному рассеиванию фазы. Тем не менее, до тех пор, пока модификация фазы в достаточной мере мала, может быть достигнуто скрытие, неощутимое на слух. Разумеется, модификация считается малой по отношению к конкретному наблюдателю, поскольку специалисты по спектральному анализу способны обнаружить те изменения, которые непрофессионалу могут показаться незначительными.

Процедура фазового кодирования заключается в следующем:

**1.** Звуковая последовательность $\boldsymbol{S}[\boldsymbol{i}]$, $(1 \le \boldsymbol{i} \le \boldsymbol{I})$ разбивается на серию $\boldsymbol{N}$ коротких сегментов (блоков) $\boldsymbol{S_n}[\boldsymbol{i}]$, $(1 \le \boldsymbol{n} \le \boldsymbol{N})$ — рис. 5.66 *а*, *б*.

**2.** К $\boldsymbol{n}$-му сегменту сигнала $\boldsymbol{S_n}[\boldsymbol{i}]$ применяется $\boldsymbol{K}$-точечное ДПФ, где $\boldsymbol{K} = \boldsymbol{I}/\boldsymbol{N}$, и создаются массивы фаз $\phi_{\boldsymbol{n}}(\omega_{\boldsymbol{k}})$ и амплитуд $\boldsymbol{A_n}(\omega_{\boldsymbol{k}})$ для $1 \le \boldsymbol{k} \le \boldsymbol{K}$ (рис. 5.66, *в*).

**3.** Запоминается разница фаз между каждыми соседними сегментами для $1 < \boldsymbol{n} \le \boldsymbol{N}$ (рис. 5.66, *г*):

$$\Delta\phi_{\boldsymbol{n}}(\omega_{\boldsymbol{k}}) = \phi_{\boldsymbol{n}}(\omega_{\boldsymbol{k}}) - \phi_{\boldsymbol{n}-1}(\omega_{\boldsymbol{k}}); \quad \Delta\phi_1(\omega_{\boldsymbol{k}}) = 0. \tag{5.59}$$

**4.** Двоичная последовательность данных представляется как $\phi_{\boldsymbol{data}} = \pi/2$ или $\phi_{\boldsymbol{data}} = -\pi/2$, отображая, соответственно, «1» или «0» (рис. 5.66, *д*): $\phi'_1(\omega_{\boldsymbol{k}}) = \phi_{\boldsymbol{data}}$.

**5.** С учетом разницы фаз воссоздается новый массив фаз для $\boldsymbol{n} > 1$ (рис. 5.66, *е*).

$$\left\| \begin{array}{c} \phi'_1(\omega_{\boldsymbol{k}}) = \phi_{\boldsymbol{data}} \\ \phi'_2(\omega_{\boldsymbol{k}}) = \phi'_1(\omega_{\boldsymbol{k}}) + \Delta\phi_2(\omega_{\boldsymbol{k}}) \\ \cdots \\ \phi'_{\boldsymbol{n}}(\omega_{\boldsymbol{k}}) = \phi'_{\boldsymbol{n}-1}(\omega_{\boldsymbol{k}}) + \Delta\phi_{\boldsymbol{n}}(\omega_{\boldsymbol{k}}) \\ \cdots \\ \phi'_{\boldsymbol{N}}(\omega_{\boldsymbol{k}}) = \phi'_{\boldsymbol{N}-1}(\omega_{\boldsymbol{k}}) + \Delta\phi_{\boldsymbol{N}}(\omega_{\boldsymbol{k}}) \end{array} \right\|. \tag{5.60}$$

**6)** Восстановление звукового сигнала осуществляется путем применения операции обратного ДПФ к исходной матрице амплитуд и модифицированной матрице фаз (рис. 5.66, *ж*, *з*).

Перед процессом расшифровывания должна быть проведена синхронизация последовательности. Приемной стороне должны быть известны длина сегмента, точки ДПФ и интервал данных. Значение основной фазы первого сегмента определяется как «0» или «1», которые представляют закодированную двоичную последовательность.

Поскольку фаза $\phi'_1(\omega_{\boldsymbol{k}})$ является измененной, соответствующим образом будут изменены и абсолютные фазы последующих сегментов. Однако относительная разность фаз между каждыми смежными сегментами будет сохранена. Именно к этой относительной разности в фазе и обнаруживается наибольшая чувствительность человеческого слуха.

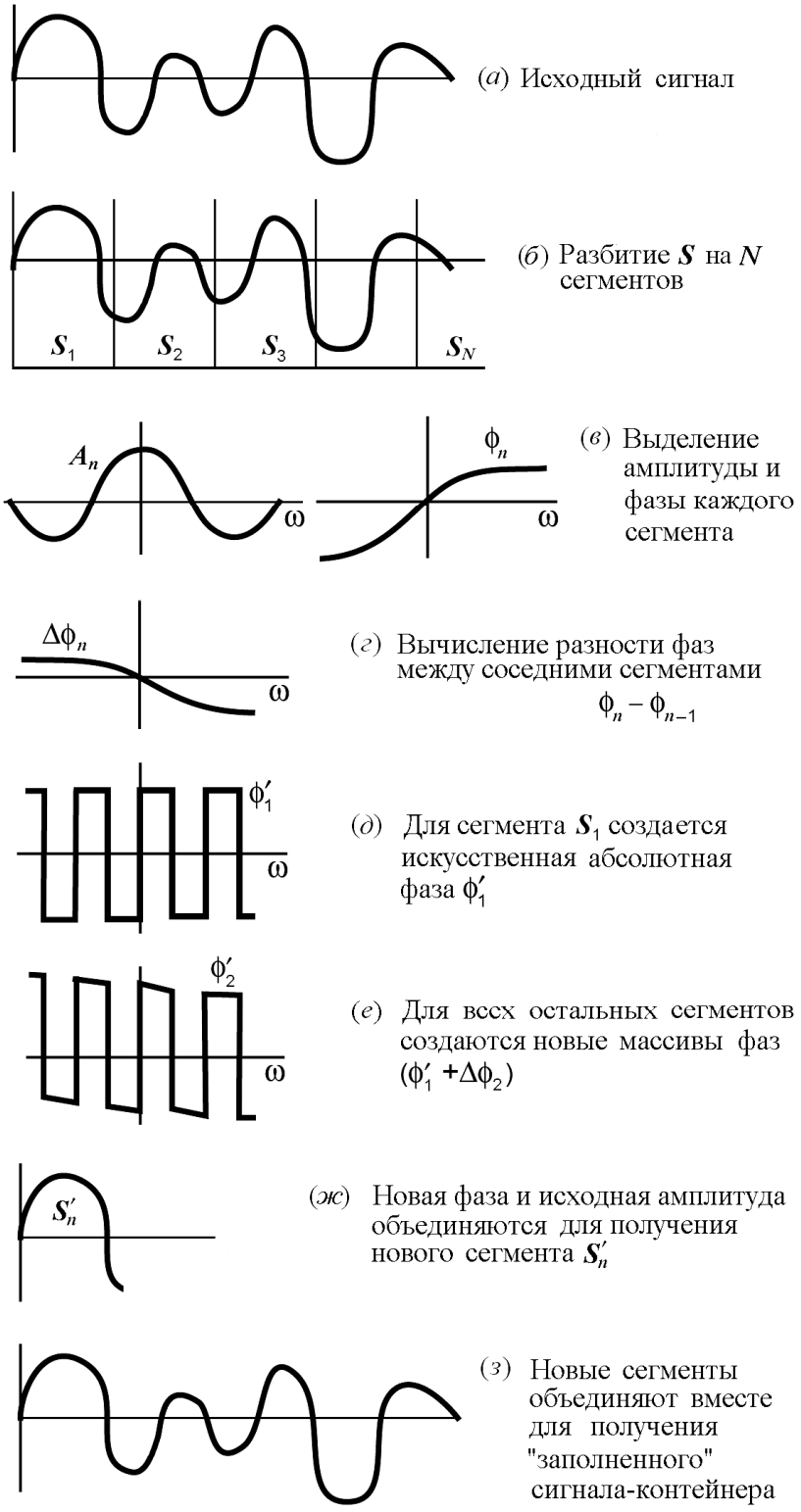



**Рис. 5.66. Блок-схема фазового кодирования**

Разброс фаз — искажение, вызванное нарушением корреляции фаз между каждой из частотных составляющих. Уменьшение разброса фаз ограничивает скорость передачи данных при фазовом кодировании. Одной из причин разброса фаз можно назвать замещение фазы $\phi_1'(\omega_k)$ двоичным кодом. Для уменьшения искажений, значение модифицированной фазы должно быть близким к ее первичному значению. А для того чтобы снизить чувствительность встроенных данных к шуму, должна увеличиваться разность между структурами модифицированной фазы. Для этого, например, биту «0» может отвечать значение $-\pi/2$, а биту «1» — значение $+\pi/2$.

Еще одним источником искажения является скорость изменения модифицированной фазы. Если искажение применено к каждому элементу дискретизации ДПФ, это с большой вероятностью разрушит связи между фазами соседних частотных составляющих, что, в результате, приведет к наложению фонового биения. Путем более медленного изменения фазы и согласования переходов между изменениями фазы, достигается существенное снижение ощутимых на слух искажений.

На рис. 5.67 изображены резкие переходы в сравнении со сглаженными. Крутые фронты фазовых переходов вызывают значительные искажения контейнера (*а*); уровень искажения уменьшается, если фронты были предварительно сглажены (*б*).

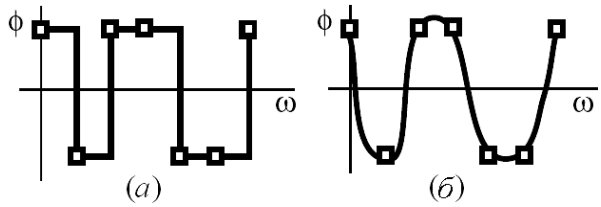



**Рис. 5.67. Сравнение резких и сглаженных переходов**

Следует отметить, что в обоих случаях, изображенных на рис. 5.67, информационные точки соответствуют одним и тем же значениям частоты. Такое плавное изменение характеризуется, к сожалению, таким недостатком, как сокращение полосы пропускания, поскольку для того чтобы сделать возможным плавный переход, необходимо зарезервировать достаточно места между каждой информационной точкой.

Недостатком схемы, построенной на основе метода фазового кодирования, является низкая пропускная способность. В экспериментах [14] ПС канала варьировалась от 8 до 32 бит/с в зависимости от звукового контекста.

Нами предлагается следующая реализация метода фазового кодирования.

**Шаг 1**

Импортируем файл исходного аудиоконтейнера в массив квантовых амплитуд дискретных отсчетов документа MathCAD:

**C** := **READWAV**("C.wav").

Скрываемые данные будем вносить в первый канал данного звукового файла: **S** := $\mathbf{C}^{\langle 1\rangle}$.

Количество элементов в указанном канале-контейнере: **I** := **rows**(**S**), **I** = 20191; **i** := 1..**I**. Временн*ая* диаграмма первого канала ИКМ сигнала изображена на рис. 5.63.

**Шаг 2**

Допустим, скрытию подлежит сообщение следующего содержания:

**M** := "© Пузыренко А.Ю., 2005 г.".

Битовая длина сообщения: $\mathbf{L_M} := 8 \cdot \mathbf{strlen}(\mathbf{M})$, $\mathbf{L_M} = 200$ бит.

**Шаг 3**

Проведем разбивание звуковой последовательности на сегменты. Количество сегментов **N** определяется длиной **K** отдельного сегмента. Как будет в дальнейшем показано, параметр **K** должен быть результатом возведения двойки в степень $\nu$, где $\nu$ – целое число, зависящее от длины $\mathbf{L_M}$ скрываемого сообщения. При этом биты информации будут заноситься в массив фаз, полученный в результате вычисления для сегмента быстрого преобразования Фурье (БПФ).

Отмеченный массив имеет размерность вдвое меньшую размерности сегмента, для которого проводилось вычисление. Следовательно, значение параметра $\nu$ можно найти из решения неравенства $2^{\nu} \ge 2 \cdot \mathbf{L_M}$, откуда

$$\nu := \mathbf{ceil}(\mathbf{log}(\mathbf{L_M}, 2) + 1),$$

где $\mathbf{log}(z, 2)$ — стандартная запись вычисления логарифма по основанию 2 от аргумента $z$; функция $\mathbf{ceil}(x)$ возвращает наименьшее целое, которое превышает или равняется аргументу $x$.

Результатом решения при $\mathbf{L_M} = 200$ будет $\nu = 9$. Следовательно, $\mathbf{K} := 2^{\nu+1}$ (степень увеличена на единицу для уменьшения искаженности контейнера после внедрения в него скрываемой информации). При этом $\mathbf{K} = 1024$; $\mathbf{k} := 1..\mathbf{K}$.

Определим количество сегментов **N**, на которое необходимо разбить последовательность аудиоданных: $\mathbf{N} := \mathbf{ceil}(\mathbf{I}/\mathbf{K})$, $\mathbf{N} = 20$; $\mathbf{n} := 1..\mathbf{N}$; $\mathbf{I}/\mathbf{K} = 19{,}718$.

Если целое значение **N** превышает результат отношения $\mathbf{I}/\mathbf{K}$, то контейнер необходимо расширить. Например, путем дописывания в конец вектора-сигнала нулей — (М.109).

$$\mathbf{S} := \left| \begin{array}{l} \textbf{for}\ \ \mathbf{i} \in 1..\mathbf{N} \cdot \mathbf{K} \\ \quad \left| \begin{array}{l} \mathbf{S_i} \leftarrow \mathbf{S_i} \ \ \textbf{if}\ \ \mathbf{i} \le \mathbf{I} \\ \mathbf{S_i} \leftarrow 0 \ \ \textbf{if}\ \ \mathbf{i} > \mathbf{I} \end{array} \right. \\ \mathbf{S} \end{array} \right. \qquad \text{(М.109)}$$

Таким образом, новое значение $\mathbf{I} := \mathbf{rows}(\mathbf{S})$, $\mathbf{I} = 20480$. Окончательно определившись с основными размерностями, проведем разбиение первичной звуковой последовательности **S**, используя программный модуль (М.110).

$$\mathbf{s} := \left| \begin{array}{l} \textbf{for}\ \ \mathbf{n} \in 1..\mathbf{N} \\ \quad \mathbf{s_n} \leftarrow \mathbf{submatrix}[\mathbf{S}, (\mathbf{n}-1)\cdot\mathbf{K}+1, \mathbf{n}\cdot\mathbf{K}, 1, 1] \\ \mathbf{s} \end{array} \right. \qquad \text{(М.110)}$$

Схема разбиения приведена на рис. 5.68.

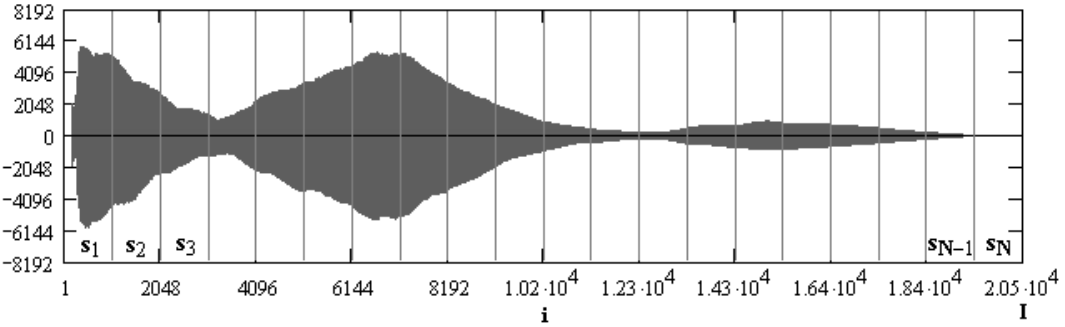



**Рис. 5.68. Сигнал $S_i$ (i = 1..I), разбитый на N сегментов $s_n$ (n = 1..N)**

**Шаг 4**

С помощью встроенной в MathCAD функции **FFT(V)** выполняем БПФ для данных, записанных в векторе-аргументе **V**. Последний должен содержать $2^{\nu}$ элементов, где $\nu$ — целое число. Результатом выполнения функции является вектор размерностью $(2^{\nu-1} + 1)$.

В нашем случае, в качестве аргумента функции БПФ будут выступать векторы отдельных сегментов $\mathbf{s_n}$ (М.111 *а*).

$$\sigma := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad \sigma_n \leftarrow \text{FFT}(s_n) \\ \sigma \end{array} \right. \qquad A := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad A_n \leftarrow \overrightarrow{|\sigma_n|} \\ A \end{array} \right. \qquad \phi := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad \phi_n \leftarrow \overrightarrow{\arg(\sigma_n)} \\ \phi \end{array} \right. \quad \text{(М.111)}$$

(*а*) (*б*) (*в*)

Каждый **n**-й элемент полученного в результате массива $\sigma$ содержит подмассив из $[(K/2) + 1]$ элементов, которые представляют собой продукт вычисления БПФ для сегмента $\mathbf{s_n}$.

Используя известные зависимости, получаем массив амплитуд и фаз — (М.111 *б*, *в*). Запись $\overrightarrow{\psi(V)}$ означает операцию векторизации — выполнение заданной операции $\psi$ для всех элементов массива **V**.

В качестве примера, приведем результат вычисления (первые 15 элементов) для 1-го сегмента (рис. 5.69).

$s_1 =$

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

$\sigma_1 =$

| | 1 |
|---|---|
| 1 | -16.171 |
| 2 | -16.039-0.37i |
| 3 | -16.548-0.428i |
| 4 | -11.696+0.936i |
| 5 | -22.515+1.469i |
| 6 | -18.979-1.948i |
| 7 | -18.442-2.908i |
| 8 | -18.062-2.386i |
| 9 | -13.601-1.661i |
| 10 | -21.488-11.192i |
| 11 | -21.555-1.861i |
| 12 | -16.348-1.132i |
| 13 | -17.928-7.227i |
| 14 | -20.536-6.973i |
| 15 | -22.17-4.883i |

$A_1 =$

| | 1 |
|---|---|
| 1 | 16.171 |
| 2 | 16.043 |
| 3 | 16.554 |
| 4 | 11.733 |
| 5 | 22.562 |
| 6 | 19.078 |
| 7 | 18.670 |
| 8 | 18.219 |
| 9 | 13.702 |
| 10 | 24.228 |
| 11 | 21.635 |
| 12 | 16.387 |
| 13 | 19.329 |
| 14 | 21.687 |
| 15 | 22.701 |

$\phi 1 =$

| | 1 |
|---|---|
| 1 | 3.142 |
| 2 | -3.118 |
| 3 | -3.116 |
| 4 | 3.062 |
| 5 | 3.076 |
| 6 | -3.039 |
| 7 | -2.985 |
| 8 | -3.010 |
| 9 | -3.020 |
| 10 | -2.661 |
| 11 | -3.055 |
| 12 | -3.072 |
| 13 | -2.758 |
| 14 | -2.814 |
| 15 | -2.925 |

$\overrightarrow{(A_1 \cdot \exp(j \cdot \phi_1))} =$

| | 1 |
|---|---|
| 1 | -16.171 |
| 2 | -16.039-0.37i |
| 3 | -16.548-0.428i |
| 4 | -11.696+0.936i |
| 5 | -22.515+1.469i |
| 6 | -18.979-1.948i |
| 7 | -18.442-2.908i |
| 8 | -18.062-2.386i |
| 9 | -13.601-1.661i |
| 10 | -21.488-11.192i |
| 11 | -21.555-1.861i |
| 12 | -16.348-1.132i |
| 13 | -17.928-7.227i |
| 14 | -20.536-6.973i |
| 15 | -22.17-4.883i |

**Рис. 5.69. Пример вычисления программных модулей (М.110) и (М.111)**

На рис. 5.70 представлена графическая интерпретация амплитудного и фазового массивов для первого сегмента.

**Шаг 5**

С помощью программного модуля (М.112) сохраняем информацию о разнице фаз между каждыми соседними сегментами.

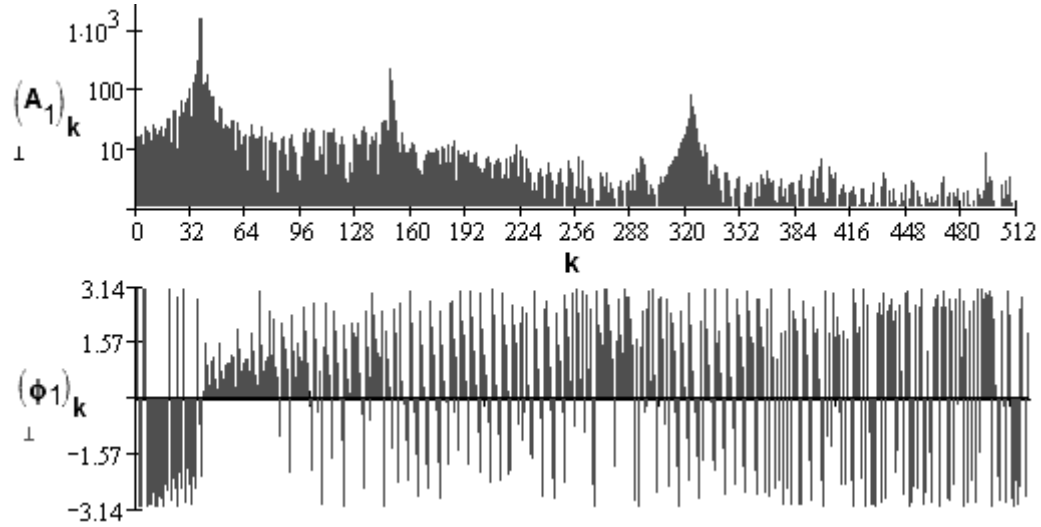



**Рис. 5.70. Пример выделенного амплитудного и фазового спектра для сегмента $s_1$**

$$\Delta\phi := \left| \begin{array}{l} \Delta\phi_1 \leftarrow 0 \cdot \phi_1 \\ \text{for } n \in 2..N \\ \quad \Delta\phi_n \leftarrow \phi_n - \phi_{n-1} \\ \Delta\phi \end{array} \right. \qquad \text{(M.112)}$$

На рис. 5.71 проиллюстрирован результат вычисления разности фаз между вторым и первым сегментами.

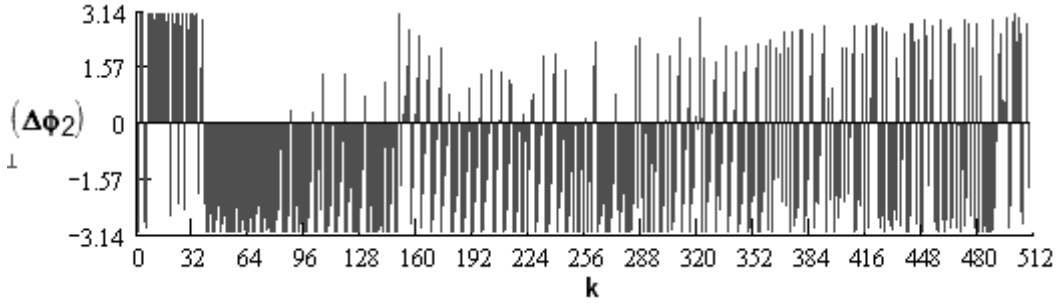



**Рис. 5.71. Пример вычисления разности фаз между сегментами $s_2$ и $s_1$**

**Шаг 6**

С помощью программного модуля (M.113), учитывая разность фаз $\Delta\phi$, воссоздаем новую матрицу фаз. При этом двоичную последовательность, в которую предварительно преобразовываем скрываемые данные, встраиваем как значение фазы, которое равняется $\pi/2$, если скрывается бит «1» и $-\pi/2$, если скрывается бит «0». Результат вычисления модуля, на примере новых фаз сегментов $s_1$ и $s_2$, приведен на рис. 5.72.

При формировании массива новых фаз внесение скрываемых данных начинается с высокочастотных составляющих. Кроме того, остаются без изменений первая и последняя составляющие полученного фазового спектра, поскольку их модификация приводит к значительному искажению исходного сигнала.

После внесения данных, которые необходимо было скрыть, массив дописывается элементами из первичного фазового массива.

```
φ' := | m ← D2B(str2vec(M)_1)
      | for τ ∈ 2.. strlen(M)                                  if strlen(M) > 1
      |     m ← stack(m, D2B(str2vec(M)_τ))
      | for k ∈ 1.. K/2 + 1
      |     if k ≤ rows(m) + 1
      |         φ'data_{K/2+1} ← (φ1)_{K/2+1}                  if k = 1 ∨ k = K/2 + 1
      |         if 1 < k ≤ K/2
      |             φ'data_{K/2+2−k} ← −π/2                    if m_{k−1} = 1
      |             φ'data_{K/2+2−k} ← π/2                     if m_{k−1} = 0
      |         φ'data_{K/2+2−k} ← (φ1)_{K/2+2−k}              if k > rows(m) + 1
      | φ'1 ← φ'data
      | for n ∈ 2.. N
      |     φ'n ← φ'n−1 + Δφn
      | φ'
```

(M.113)

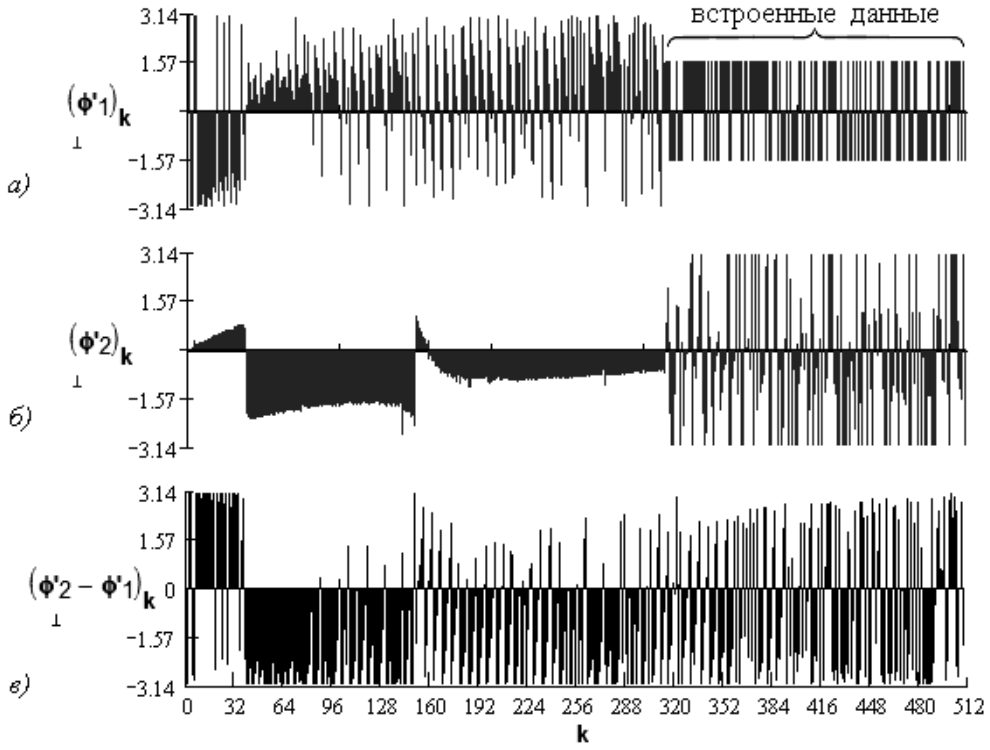



**Рис. 5.72. Результаты встраивания данных в фазовый спектр сегмента $s_1$ (*а*), вычисления новых значений фаз спектра сегмента $s_2$ (*б*), проверки разности фаз между спектрами сегментов $s_2$ и $s_1$ (*в*)**

**Шаг 7**

Восстанавливаем сегменты, путем применения обратного БПФ (ОБПФ) к **n** исходным массивам амплитуд ($A_n$) и модифицированным массивам фаз ($\phi'_n$) — программный модуль (М.114). При этом используем встроенную функцию ОБПФ вида **IFFT(W)**, где вектор **W** должен содержать [(**K**/2) + 1] элементов. Результатом выполнения функции является вектор **V**, количество элементов в котором равняется **K**.

$$s' := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad \left| \begin{array}{l} \sigma \leftarrow \overrightarrow{\left(A_n \cdot \exp\left(j \cdot \phi'_n\right)\right)} \\ s'_n \leftarrow \text{IFFT}(\sigma) \end{array} \right. \\ s' \end{array} \right. \qquad \text{(М.114)}$$

Восстановленные сегменты объединяем в общий массив — (М.115).

$$S' := \left| \begin{array}{l} S' \leftarrow s'_1 \\ \text{for } n \in 2..N \\ \quad S' \leftarrow \text{stack}\left(S', s'_n\right) \\ S' \end{array} \right. \qquad \text{(М.115)}$$

Результат объединения приведен на рис. 5.73 (напомним, что в данном случае нами было скрыто 200-битовое сообщение).

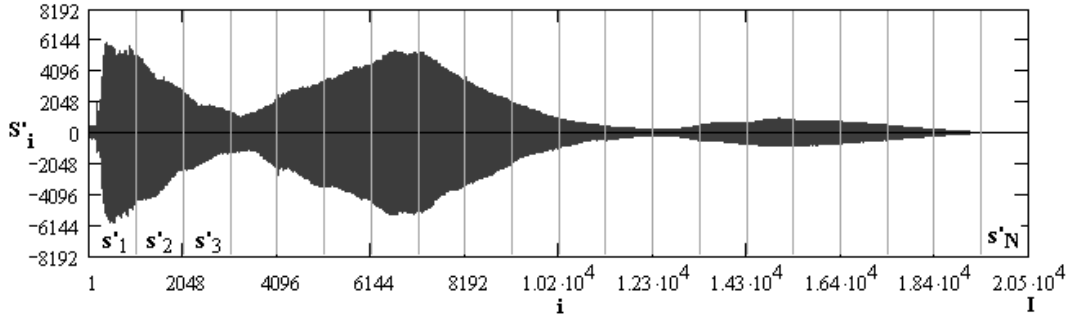



**Рис. 5.73. Сигнал, восстановленный по сегментам $s'_n$ (n = 1..N) при объеме скрытого сообщения 200 бит**

На рис. 5.74 изображен восстановленный звуковой сигнал, в случае скрытия в нем 512-битового сообщения при сохранении неизменной длины сегмента (**K** = 1024). Очевидно, что в данном случае уровень скрытости конфиденциального сообщения является недопустимо низким.

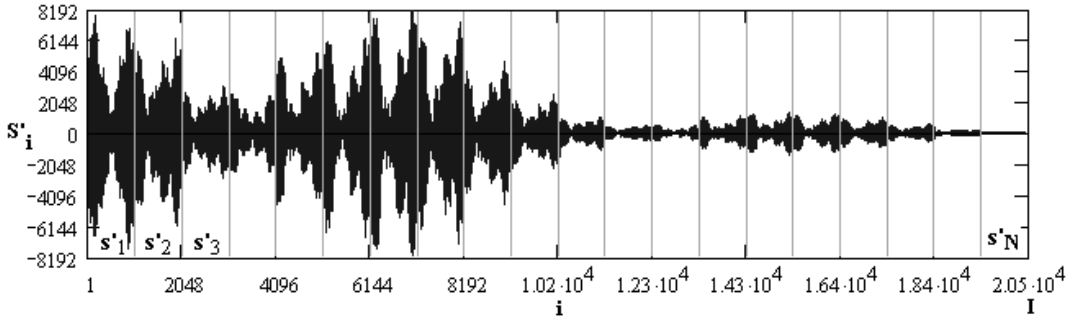



**Рис. 5.74. Сигнал, восстановленный по сегментам $s'_n$ (n = 1..N) при объеме скрытого сообщения 512 бит**

**Шаг 8**

Перед записью звукового файла необходимо увеличить размерность вектора, который соответствует немодифицированному каналу (в нашем случае — второму), до размерности модифицированного канала (первого). Для этого можно воспользоваться модулем (М.116), который при необходимости дописывает нули в конец исходного массива.

$$\mathbf{C2} := \left| \begin{array}{l} \textbf{for}\ \ i \in 1 .. \mathbf{rows}(S') \\ \quad \left| \begin{array}{l} C2_i \leftarrow \left(C^{\langle 2 \rangle}\right)_i \ \ \textbf{if}\ \ i \le \mathbf{rows}\left(C^{\langle 2 \rangle}\right) \\ C2_i \leftarrow 0 \ \ \textbf{if}\ \ i > \mathbf{rows}\left(C^{\langle 2 \rangle}\right) \end{array} \right. \\ C2 \end{array} \right. \qquad \text{(М.116)}$$

В результате появляется возможность объединить массивы обоих каналов в общий массив: **CM** := **augment**(**S'**, **C2**) и, в конечном итоге, провести запись в файл:

**WRITEWAV**("CM_Phase.wav", $\mathbf{f_д}$, **Q**) := **CM**.

**Шаг 9**

Рассмотрим алгоритм извлечения скрытой информации.

На основе аудиофайла, представляющего собой заполненный стеганоконетйнер, формируем массив квантованных амплитуд дискретных отсчетов:

**CM*** := **READWAV**("CM_Phase.wav").

Получателю должны быть известны: длина каждого из сегментов (в нашем случае — **K*** := 1024); точки БПФ, в которые встраивались данные; алгоритм встраивания данных (канал, в который вносилась информация, метод ее внесения и т.п.).

Пусть известно, что скрытые данные могут содержаться в первом канале. Следовательно, **S*** := **CM***$^{\langle 1 \rangle}$. Общее количество элементов в указанном контейнере: **l*** := **rows**(**S***), **l*** = 20480. Количество сегментов, на которое необходимо разделить контейнер: **N*** := **l***/**K***, **N*** = 20.

В соответствии с программным модулем (М.110) проводим разбитие звуковой последовательности **S*** на сегменты $\mathbf{s^*_n}$.

Для первого сегмента вычисляем БПФ и определяем массив фаз: $\sigma^*_1$ := **FFT**($\mathbf{s}^*_1$); $\phi^*_1 := \overrightarrow{\mathbf{arg}\left(\sigma^*_1\right)}$.

Для извлечения скрытой информации из полученного массива фаз $\phi^*_1$ используем программный модуль (М.117).

В данном модуле последовательно анализируются значения фаз каждой частотной составляющей спектра первого сегмента на факт отклонения от установленного порога (как в сторону отрицательных, так и в сторону положительных значений). В качестве значения порога не рекомендуется использовать строгое равенство $\pm\pi/2$, поскольку аудиофайл в процессе передачи может претерпеть определенные изменения. Кроме того, операции БПФ возвращают значения, округленные в некотором приближении. Исходя из этого, для «смягчения» критериев поиска порог рекомендуется несколько снизить (в нашем случае был установлен уровень $\pm\pi/3$). При значении фазы, меньшем отрицательного значения порога, принимается решение о скрытом бите «1», в случае значения фазы, которое превышает положительный порог, делается вывод о скрытии бита «0».

```
M* := | for τ ∈ 1 .. K*/2
      |     d ← (ϕ*1)_(rows(ϕ*1)−τ)
      |     b_τ ← 1  if  d < −π/3
      |     b_τ ← 0  if  d > π/3
      |     break  otherwise
      | for i ∈ 1 .. floor(rows(b)/8)
      |     B ← submatrix[b , (i − 1)·8 + 1 , i·8 , 1 , 1]
      |     M*_i ← B2D(B)
      | vec2str(M*)
```

(M.117)

Если анализируемый аудиофайл содержит скрытую фазовым методом информацию, результатом выполнения модуля (М.117) будет строка символов: **M* = M**.

Рассчитанные показатели звукового искажения сведены в табл. 5.6.

### *5.4.3. Метод расширения спектра (временная область)*

В стандартном канале связи нередко является желательным сосредоточить информацию в как можно более узком диапазоне частотного спектра, например, для того чтобы сохранить имеющуюся полосу пропускания и уменьшить мощность сигнала. С другой стороны, основной метод расширения спектра предназначен для шифрования потока информации путем «рассеивания» кодированных данных по всему возможному частотному спектру. Последнее делает возможным прием сигнала даже при наличии помех на определенных частотах.

В [14] рассматривается технология расширения спектра сигнала прямой последовательностью (РСПП). Как уже указывалось выше (см. методы скрытия данных в изображении путем расширения спектра), методы РСПП расширяют сигнал данных (сообщения), умножая его на элементарную посылку — ПСП максимальной длины, модулированную известной частотой.

Поскольку аудиосигналы, используемые в качестве контейнеров, имеют дискретный формат, то для кодирования в качестве частоты элементарной посылки можно использовать частоту дискретизации. Как следствие, дискретный характер сигнала устраняет наиболее сложную проблему, которая возникает при получении сигнала с расширенным прямой последовательностью спектром, — корректного определения начала и конца составляющих элементарной посылки с целью фазовой синхронизации. Следовательно, возникает возможность использования намного более высокой частоты следования элементарных посылок, и, таким образом, получения значительной связанной с ней скорости передачи данных. Кроме этого также могут применяться разнообразные алгоритмы блокирования сигнала, однако в вычислительном плане они являются достаточно сложными.

В РСПП для шифрования и дешифрования информации необходим один и тот же ключ — псевдослучайный шум, который в идеальном случае имеет плоскую частотную характеристику во всем диапазоне частот (так называемый белый шум).

Ключ применяется к скрываемой информации и трансформирует ее последовательность в последовательность с расширенным спектром.

Метод РСПП по отношению к аудиосигналам заключается в следующем. Сигнал данных умножается на сигнал несущей и псевдослучайную шумовую последовательность, характеризующуюся широким частотным спектром. В результате этого спектр данных расширяется на всю доступную полосу. В дальнейшем последовательность расширенных данных ослабляется и прибавляется к исходному сигналу как аддитивный случайный шум (рис. 5.75).

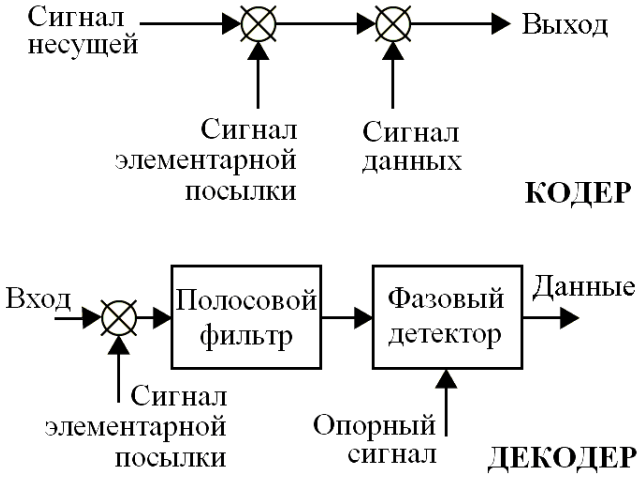



**Рис. 5.75. Структурная схема кодека с расширением спектра**

РСПП использует двоичную фазовую манипуляцию, поскольку фаза сигнала ПСП поочередно чередуется с фазой модулированной двоичной последовательности сообщения (рис. 5.76).

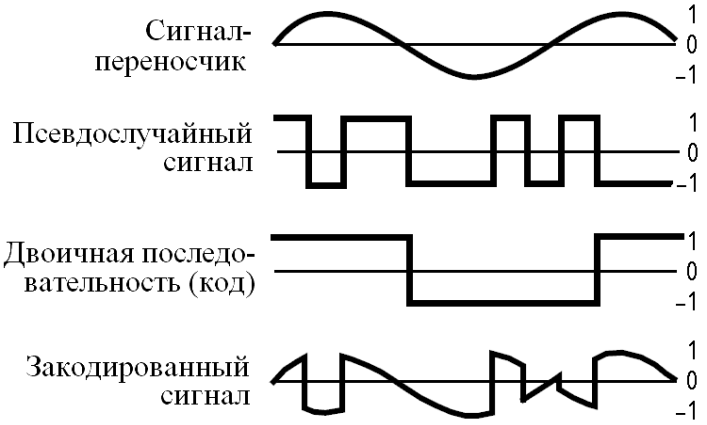



**Рис. 5.76. Информация, синтезированная расширением спектра и шифрованная методом прямой последовательности**

На стадии извлечения фазовые значения $\phi_0$ и $\phi_0 + \pi$ интерпретируются, соответственно, как биты «1» и «0», которыми кодировалась двоичная последовательность данных. При этом предусматривается следующее:

- Псевдослучайный ключ представляет собой М-последовательность (то есть он имеет максимально возможное количество комбинаций, которые равномерно распределены в заданном диапазоне, и максимально долго не повторяется). Следовательно, он имеет относительно плоский частотный спектр.

- Принимающей стороне известен поток ключей для шифрования. Выполнена синхронизация сигнала, а также известны точки начала и конца расширенных данных.
- Принимающей стороне также известны следующие параметры: частота следования элементарных посылок, скорость передачи данных и частота (вид) несущей.

В отличие от рассмотренного выше фазового кодирования, метод РСПП вводит в звук аддитивный случайный шум. Для того чтобы держать уровень шума низким и неощутимым на слух, расширенный код ослабляется (без адаптации) приблизительно до уровня 0,5% от динамического диапазона звукового файла-контейнера.

Объединение несложной техники повторения и кодирования с исправлением ошибок позволяет гарантировать целостность двоичной последовательности. Короткие сегменты двоичной кодовой комбинации объединяются и складываются с сигналом аудиоконтейнера таким образом, чтобы уменьшить шумы переходных процессов. Для этого проводится усреднение по всему сегменту в процессе декодирования.

Во время исследований метода РСПП, авторами [14] была получена скорость передачи данных около 4 бит в секунду.

Приведем пример реализации метода РСПП.

**Шаг 1**

Начальные данные следующие: контейнер **C** := **READWAV**("C.wav"); сообщение **M** := "© Пузыренко А.Ю., 2005 г." длиной $L_M := 8 \cdot \mathbf{strlen}(\mathbf{M})$, $L_M = 200$ бит.

В качестве контейнера будем использовать первый канал сигнала: $\mathbf{C1} := \mathbf{C}^{\langle 1 \rangle}$.

Для встраивания $L_M$-битового сообщения в контейнер, имеющий **rows**(**C1**) = = 20191 дискретных отсчетов, последний разобьем на $\mathbf{N} := \mathbf{floor}(\mathbf{rows}(\mathbf{C1}) / L_M)$, **N** = 100 сегментов, каждый из которых будет предназначен для встраивания одного бита сообщения.

**Шаг 2**

Для каждого бита сообщения необходимо сгенерировать ПСП в виде последовательности +/–1, длиной, как минимум, **N** элементов. За основу генератора ПСЧ можно взять ЛРСОС, описанный в п. 5.3.3.3. Достаточное количество разрядов регистра: **d** := **ceil**(**log**(**N**+1, 2)), **d** = 7. При этом период генерированной ПСП составит $2^{\mathbf{d}} - 1 = 127 > \mathbf{N}$.

Программный модуль (М.118) позволяет получать ПСП при различных значениях **d**.

Результирующая последовательность определяется наименьшим значащим битом состояния регистра ($\mathbf{CHIP_i} \leftarrow \mathbf{R_{bin_1}}$). Руководствуясь достаточностью, процесс генерации длится до получения **N**-го бита ПСП. На заключительном этапе полученная ПСП {0, 1} преобразовывается в ПСП {–1, 1}.

**Шаг 3**

Непосредственно встраивание бит сообщения в контейнер выполняется программным модулем (М.119). В начале данного модуля строка символов **M** преобразовывается в вектор значений $\{-1, 1\}$. **m**-й элемент полученного вектора с помощью соответствующей ему ПСП **CHIP**(**m**) накладывается на **m**-й сегмент контейнера **c**.

Параметр $\alpha$ при этом выбирается исходя из требований стойкости системы и незаметности модификации носителя (рекомендуются значения $\alpha$ порядка одной сотой).

$$\mu_1 := (1)^T$$
$$\mu_2 := (1\ \ 1)^T$$
$$\mu_3 := (1\ \ 0\ \ 1)^T$$
$$\mu_4 := (0\ \ 0\ \ 1\ \ 1)^T$$
$$\mu_5 := (0\ \ 1\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_6 := (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_7 := (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_8 := (0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_9 := (0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{10} := (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{11} := (0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{12} := (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\vdots$$

```
CHIP(start) := | μ ← μ_d
               | for i ∈ 1..N
               |   if i = 1
               |     R_dec_i ← start
               |     R_bin ← D2B(R_dec_i)
               |   if i ≥ 2
               |     bit ← 0
               |     for j ∈ 1..d
               |         bit ← R_bin_j ⊕ bit  if μ_j = 1
               |     R ← R_bin
               |     for j ∈ 1..d
               |       | R_bin_j ← R_(j−1)  if j > 1
               |       | R_bin_j ← bit  if j = 1
               |     R_dec_i ← B2D(R_bin)
               |   CHIP_i ← R_bin_1
               | CHIP·2 − 1
```

(M.118)

Модифицированные сегменты **s** объединяются в общий вектор **S1**. После встраивания последнего ($L_M$-го) бита сообщения вектор **S1** удлиняется на длину контейнера **C1** конечными, не претерпевшими модификации элементами последнего. Результат встраивания изображен на рис. 5.77.

```
S1 := | M_vec ← str2vec(M)
      | M_vec_bin ← D2B(M_vec_1)
      | for j ∈ 2.. strlen(M)
      |     M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
      | M_vec_bin ← 2·M_vec_bin − 1
      | m ← 1
      | while m ≤ L_M
      |     c ← submatrix[C1, [N·(m − 1) + 1], N·m, 1, 1]
      |     s ← c + α·((c·CHIP(m))·M_vec_bin_m)
      |     S1 ← s  if m = 1
      |     S1 ← stack(S1, s)  if m > 1
      |     m ← m + 1
      | S1 ← stack(S1, submatrix(C1, L_M·N + 1, rows(C), 1, 1))
      | S1
```

(M.119)

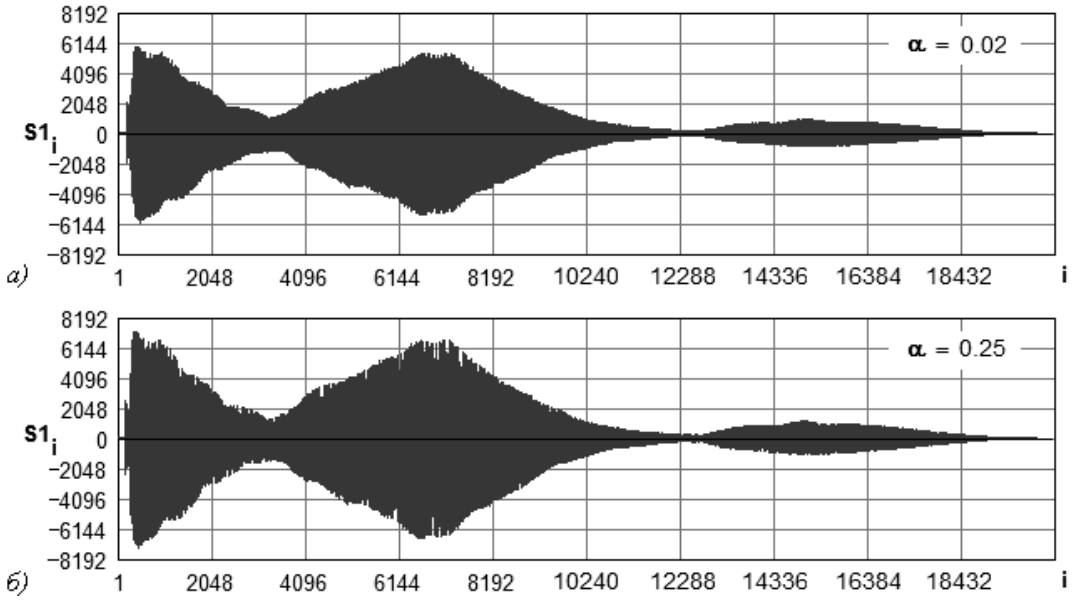



**Рис. 5.77. Временны*е* диаграммы первого канала сигнала-контейнера при внесении скрываемых данных методом РСПП при α = 0.02 (*а*) и α = 0.25 (*б*)**

Массивы первого (модифицированного) и второго каналов объединяем в общий массив: **S** := **augment**(**S1**, $\mathbf{C}^{<2>}$) и производим запись полученного результата в аудиофайл:

**WRITEWAV**("S.wav", $\mathbf{f_д}$, **Q**) := **S**.

**Шаг 4**

Процесс извлечения заключается в следующем. После импортирования аудиофайла в документ MathCAD (**S*** := **READWAV**("S.wav")) из полученного массива выделяется канал, в который было произведено встраивание (в нашем случае — первый): **S1*** := $\mathbf{S}^{\langle 1 \rangle}$.

Принимающая сторона должна иметь оригинальный аудиофайл, из которого также извлекается соответствующий канал: **C1** := $\mathbf{C}^{\langle 1 \rangle}$. Известным должно быть и количество сегментов, на которое разбивается сигнал-носитель: **N*** := 100.

Программный модуль извлечения сообщения (М.120), встроенного методом РСПП, представлен ниже.

```
M* := | m ← 1
      | while  m·N* ≤ rows(S*)
      |        | s* ← submatrix[S1*, [N*·(m − 1) + 1], N*·m, 1, 1]
      |        | c ← submatrix[C1, [N*·(m − 1) + 1], N*·m, 1, 1]
      |        | c' ← [(s* − c)·CHIP*(m)]→
      |        | M*bin_m ← 1  if  sign(mean(c)) = sign(mean(c'))
      |        | M*bin_m ← 0  if  sign(mean(c)) ≠ sign(mean(c'))
      |        | m ← m + 1
      | for  j ∈ 1 .. rows(M*bin) ÷ 8
      |        M*vec_j ← B2D(submatrix(M*bin, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*vec)
```

(М.120)

Если в сегмент было встроено «1», средние значения оригинального (**c**) и модифицированного (**c'**) сегментов будут иметь одинаковые знаки, если же «0» — разные.

Недостатком такой системы можно назвать необходимость в наличии оригинального (не модифицированного) аудиофайла у принимающей стороны. Возможным выходом из такой ситуации является, например, запись модифицированного канала вместо второгого, при этом оригинальный первый канал остается без изменений: **S** := **augment**($\mathbf{C}^{\langle 1 \rangle}$, **S1**). Таким образом, одновременно будут передаваться как пустой, так и заполненный контейнеры. Отмеченная возможность встраивания учитывается внесением следующих изменений в модуль (М.120):

```
| s* ← submatrix[S*⟨2⟩, [N*·(m − 1) + 1], N*·m, 1, 1]
| c ← submatrix[S*⟨1⟩, [N*·(m − 1) + 1], N*·m, 1, 1]
|  ⋮
```

(М.121)

Разумеется, модифицировать указанным образом весь стереофонический аудиосигнал нецелесообразно из-за полного исчезновения стереоэффекта. Поэтому можно ограничиться одним или несколькими заранее обусловленными с принимающей стороной участками или даже отдельными отсчетами на протяжении всего сигнала (например, договориться об использовании алгоритма псевдослучайного

выбора). На основании указанных участков (отсчетов) создается выборка, в которую или из которой и будет выполняться встраивание или же извлечение бит сообщения.

Вычисленные показатели звукового искажения аудиоконтейнера в результате встраивания сообщения с расширенным спектром занесено в табл. 5.6.

### *5.4.4. Скрытие данных с использованием эхо-сигнала*

Данный метод подразумевает под собой встраивание данных в аудиосигнал-контейнер путем введения в него эхо-сигнала [14]. Данные скрываются изменением трех параметров эхо-сигнала: начальной амплитуды, скорости затухания и сдвига (рис. 5.78).

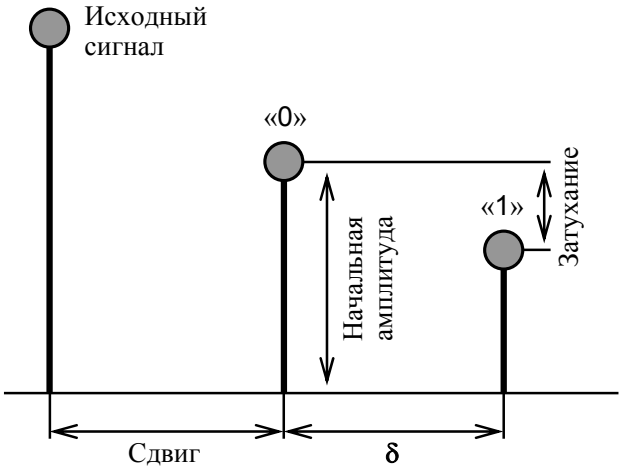



**Рис. 5.78. Регулируемые параметры эхо-сигнала**

Когда сдвиг (или задержка) между первичным и эхо-сигналом уменьшается, два сигнала смешиваются. Начиная с некоторого значения задержки, ССЧ становится не способной обнаружить разницу между двумя сигналами, а эхо-сигнал воспринимается только как дополнительный резонанс. Упомянутое значение трудно определить точно, поскольку оно зависит от качества первичной звукозаписи, типа звука, для которого формируется эхо-сигнал, и, в конечном итоге, от слушателя.

В общем случае, авторы [14] пришли к выводу, что для большинства звуков и большинства слушателей смешивание происходит при задержке, соответствующей приблизительно одной миллисекунде.

Стеганокодер использует два времени задержки: одно для представления двоичного нуля («сдвиг» на рис. 5.78), а другое — для представления двоичной единицы («сдвиг + $\delta$»). Оба времени задержки являются меньшими того предельного времени, за которое ССЧ способна распознать эхо-сигнал. Кроме уменьшения времени задержки для обеспечения неощущаемости также можно установить уровни начальной амплитуды и времени затухания, которые бы не превышали порог чувствительности ССЧ.

Процесс встраивания данных может быть представлен в виде устройства, которое реализует одну из двух возможных системных функций. Во временно*й* области системные функции — это дискретные во времени экспоненты (рис. 5.79), отличие между которыми состоит лишь в задержке между импульсами.

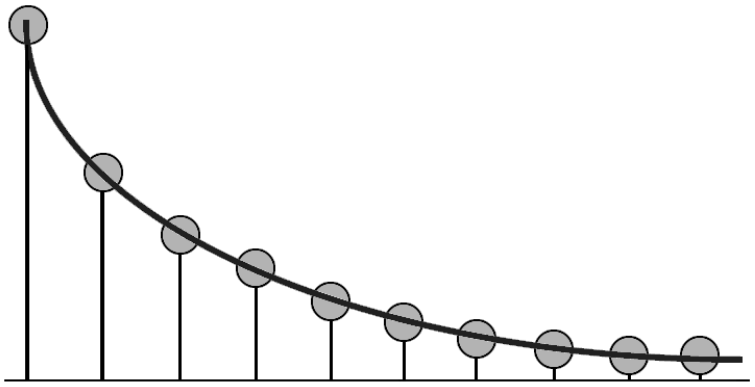


**Рис. 5.79. Дискретный во времени экспоненциал**

Рассмотрим пример с двумя импульсами (один для копирования исходного сигнала, а другой — для формирования эхо-сигнала). Очевидно, что увеличение количества импульсов приведет к возрастанию количества эхо-сигналов.

На рис. 5.80 представлены системные функции для кодирования двоичных «1» и «0». Обработка сигнала в соответствии с рис. 5.80, *а* или *б* будет иметь своим результатом закодированный сигнал (рис. 5.81).

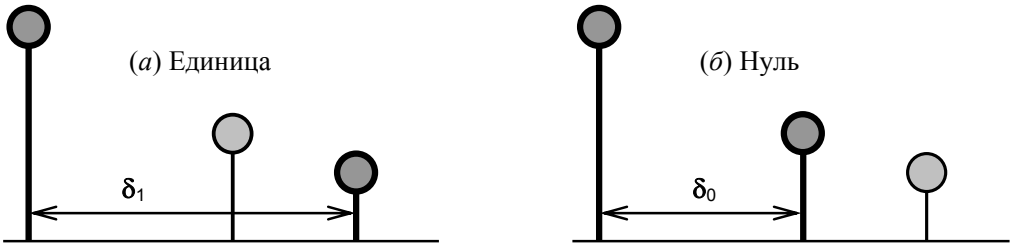



**Рис. 5.80. Представления эхо-сигнала**

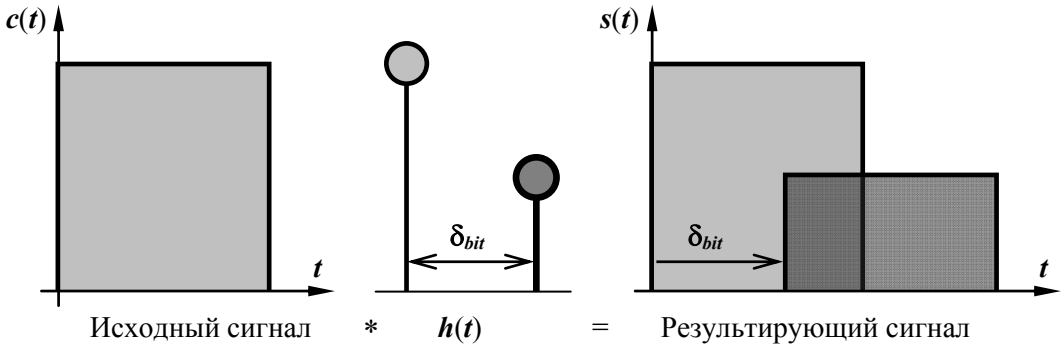



**Рис. 5.81. Пример эхо-отображения**

Задержка $\delta_{bit}$ между первичным и эхо-сигналом является зависимой от того, какое представление или системная функция была использована. Представление «1» создается задержкой в $\delta_1$ секунд, тогда как представление «0» — задержкой в $\delta_0$ секунд.

Для того чтобы в первичный сигнал закодировать более одного бита, сигнал раскладывается на меньшие сегменты. Каждый сегмент при этом рассматривается

как отдельный сигнал и в него может быть встроен (путем эхо-отображения) один бит информации. Результирующий закодированный сигнал (содержащий несколько бит) представляет собой новое объединение всех независимо закодированных сегментов исходного сигнала.

На рис. 5.82 изображен пример, при котором сигнал был разделен на 7 равных сегментов, помеченных как ***a***, ***b***, ***c***, ***d***, ***e***, ***f*** и ***g***.

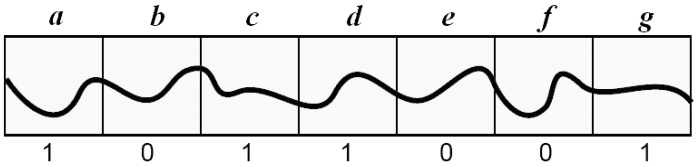



**Рис. 5.82. Разбиение первичного сигнала на меньшие сегменты для встраивания информации, представляющей собой последовательность двоичных данных**

Пусть необходимо, чтобы сегменты ***a***, ***c***, ***d*** и ***g*** содержали «1». Следовательно, для каждого из них нужно применить системную функцию представления единицы (рис. 5.80, *а*). Каждый сегмент индивидуально сворачивается с системной функцией. Нули, помещенные в сегменты ***b***, ***e*** и ***f***, кодируются аналогично, используя способ представления нуля (рис. 5.80, *б*).

Полученные после сворачивания с соответствующей функцией результаты повторно объединяются.

Для достижения минимальной заметности повторного объединения, в [14] предварительно предлагается создать отдельные «единичный» и «нулевой» эхо-сигналы, повторяя первичный и используя соответствующие представления «1» и «0». Полученные в результате сигналы изображены на рис. 5.83.

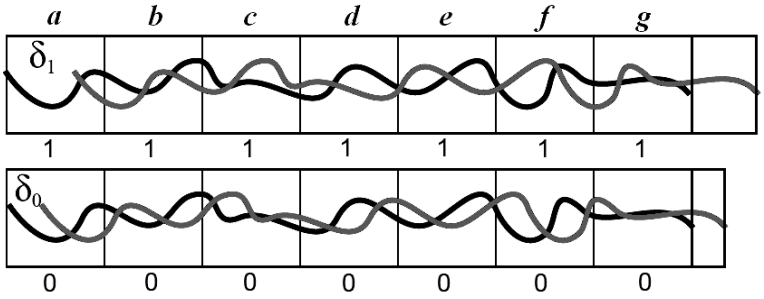



**Рис. 5.83. Создание «единичного» и «нулевого» эхо-сигналов (более светлая линия)**

«Единичный» и «нулевой» эхо-сигналы содержат, соответственно, только единицы и нули. Для того чтобы объединить эти два сигнала, также создаются два смешивающихся сигнала (рис. 5.78), которые представляют собой последовательность двоичных данных, состояние которой зависит от того, какой бит необходимо скрыть в том или ином сегменте первичного сигнала.

«Единичный» и «нулевой» смешивающие сигналы умножаются на соответствующие им эхо-сигналы. Иными словами, последние масштабируются единицей, или нулем на протяжении всего времени действия сигнала в зависимости от того, какой бит предусматривается поместить в любой из его отдельных сегментов. В дальнейшем два результата складываются друг с другом.

Необходимо заметить, что «нулевой» смешивающий сигнал представляет собой инверсию «единичного». Кроме этого, фронты переходов каждого из сигналов яв-

ляются наклонными. Сума обоих смешивающих сигналов всегда равняется единице. Все это позволяет получить плавный переход между сегментами, кодированными разными битами, а также предотвращает возникновению резких изменений в звучании результирующего (смешанного) сигнала.

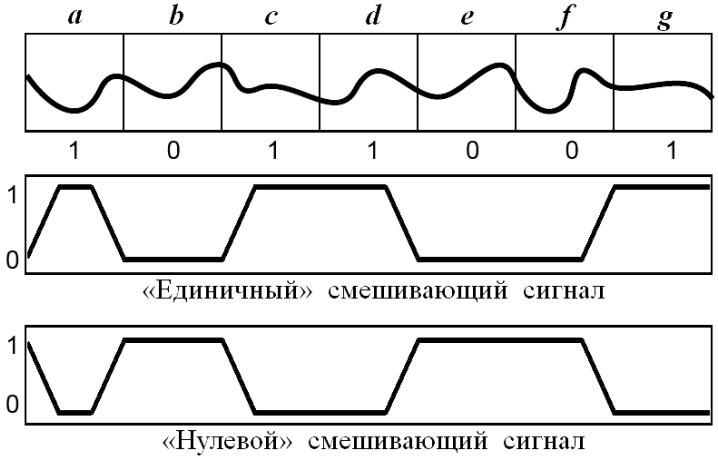



**Рис. 5.84. Смешивающие сигналы**

Блок-схема, которая отображает полный процесс встраивания, показана на рис. 5.85.

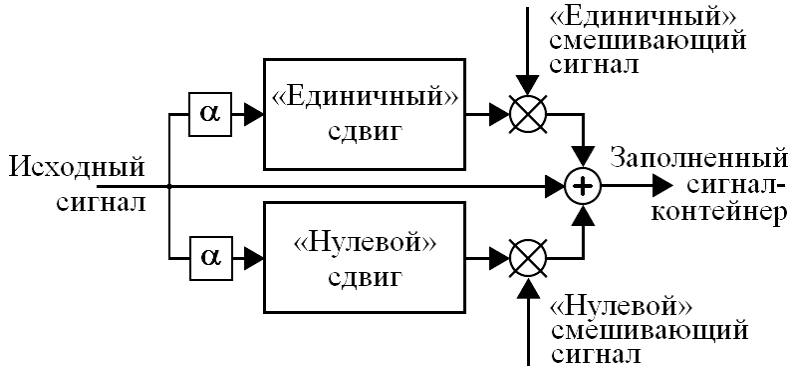



**Рис. 5.85. Процесс встраивания информации методом эхо-сигнала**

Извлечение вложенной информации подразумевает под собой выявление интервала между эхо-сигналами отдельных сегментов. Для этого необходимо исследовать в двух позициях амплитуду автокорреляционной функции (АКФ) косинус-преобразования Фурье натурального логарифма спектра мощности (или так называемого кепстра) кодированного сигнала [92, 93]:

$$\boldsymbol{AC} = \boldsymbol{F}^{-1}\left\{\left[\ln_{compl.}\left(\boldsymbol{F}(\boldsymbol{x})\right)\right]^2\right\}. \qquad (5.61)$$

Рассмотрим пример процесса извлечения, приведенный в [14]. Пусть получено закодированный сигнал, представляющий собой такую последовательность импульсов, в которой последние отделены друг от друга определенным интервалом и характеризуются экспоненциальным затуханием амплитуды. Во всех других точках сигнал равняется нулю (рис. 5.86).

Следующим шагом является поиск кепстра эхо-версии сигнала. Результат вычисления кепстра делает интервал между эхо-сигналом и первичным сигналом несколько более выраженным.

К сожалению, результат вычисления кепстра кроме всего прочего дублирует эхо-сигнал через каждые $\delta$ секунд. На рис. 5.87 это изображено наличием последовательности импульсов на выходе. Сверх того, амплитуда импульсов, которые представляют эхо-сигнал, является малой по отношению к первичному сигналу. Как следствие, их трудно обнаружить

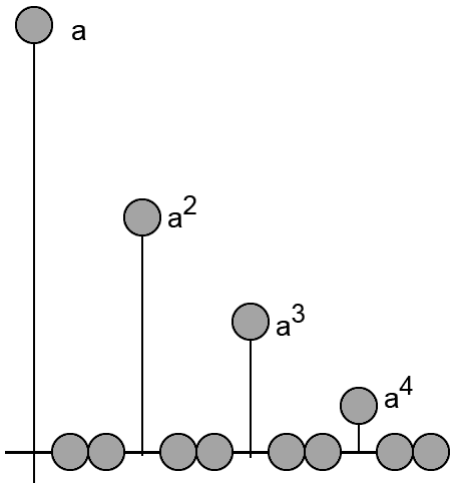



**Рис. 5.86. Пример сигнала** $x[n]= a^n \cdot u[n]$; $(0 < a < 1)$

Решение данной проблемы заключается в вычислении АКФ кепстра. С помощью однократного отображения сигнала с задержкой $\delta$ (рис. 5.88), получаем результат, изображенный на рис. 5.89. При этом значительно усилен только первый импульс, поскольку его «поддерживают» следующие за ним импульсы. Таким образом, в позиции первого импульса мы получаем всплеск. Подобно первому импульсу, всплеск повторяется через $\delta_1$ или же через $\delta_0$ секунд после всплеска первичного сигнала. Остаточные составляющие импульсов стремятся к нулю, что позволяет эффективно бороться с шумами.

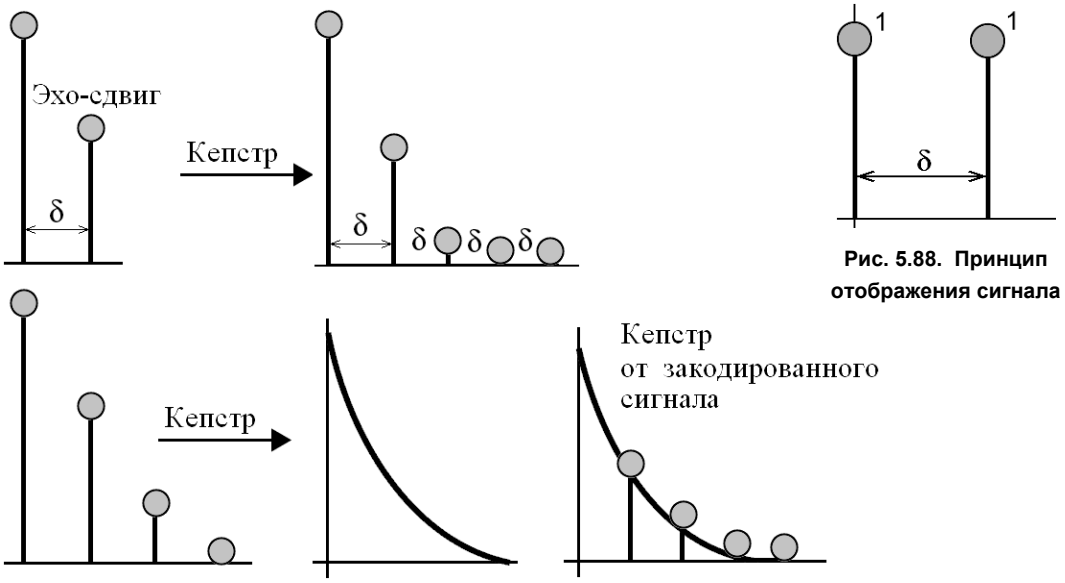



**Рис. 5.88. Принцип отображения сигнала**

**Рис. 5.87. Процесс получения кепстра от эхо-кодированного сигнала**

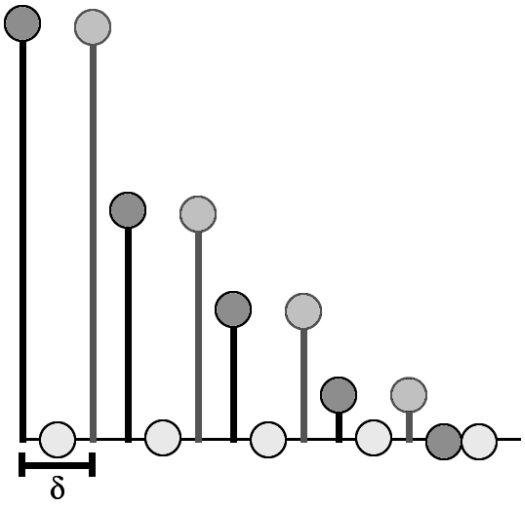



**Рис. 5.89. Результат однократного отображения сигнала**

Приведем критерий принятия решения относительно того, какой бит («1» или «0») скрыт во временно*й* задержке $\delta$ всплеска АКФ по отношению к первичному сигналу. Вспомним, что «1» кодировалась размещением эхо-сигнала через $\delta_1$, а «0» — через $\delta_0$ секунд после оригинала. Аналогично при извлечении — бит является единичным, если значение АКФ через $\delta_1$ секунд больше, чем через $\delta_0$ секунд. В противном случае бит считается нулевым.

По утверждению авторов [14], с помощью данного метода вполне возможно скрывать/извлекать информацию в виде двоичного кода в/из потока аудиоданных с минимальным изменением первичного сигнала при пропускной способности, приблизительно, в 16 бит/с. Под минимальным изменением подразумевается тот факт, что среднестатистический человек не будет способен при этом почувствовать существенную разницу между модифицированным и первичным сигналами.

Однако, в другой своей работе [108] авторы указывают на то, что предложенный ими метод не является универсальным — для некоторых аудиосигналов невозможно получить достаточно высокий коэффициент правильно распознанных при извлечении бит даже при отсутствии помех в канале связи.

Рассмотрим реализацию метода эхо-кодирования с помощью системы MathCAD.

**Шаг 1**

Исходные данные: контейнер с частотой дискретизации $\mathbf{f_д} = 22050$ Гц и количеством бит на один уровень квантования $\mathbf{Q} = 16$:

$$\mathbf{C_{total}} := \mathbf{READWAV}(\text{"C.wav"});\quad \mathbf{C} := \mathbf{C_{total}}^{\langle 1 \rangle};\quad \mathbf{rows}(\mathbf{C}) = 20191.$$

Сообщение $\mathbf{M}$ := "Опасность" длиной $\mathbf{N_M} := \mathbf{strlen}(\mathbf{M})$, $\mathbf{N_M} = 9$ символов или $\mathbf{L_M} := 8 \cdot \mathbf{N_M}$, $\mathbf{L_M} = 72$ бит.

**Шаг 2**

Пусть нулевая задержка между первичным и эхо-сигналом составляет $\delta_0 := 20$ дискретных отсчетов (или $\delta_0 / \mathbf{f_д} = 0.907$ мс), а единичная — $\delta_1 := 30$ отсчетов (или $\delta_1 / \mathbf{f_д} = 1.361$ мс).

«Единичный» и «нулевой» эхо-сигналы получим с помощью простого смещения на $\delta_{0(1)}$ отсчетов элементов контейнера-оригинала и поэлементного суммирования полученных векторов (предварительно умноженных на коэффициент затухания $\alpha$) с вектором **C** — программные модули (М.122) и (М.123).

```
C0 := | for i ∈ 1 .. δ0
      |     C0_i ← 0
      | C0 ← α·stack(C0 , C)

C1 := | for i ∈ 1 .. δ1
      |     C1_i ← 0
      | C1 ← α·stack(C1 , C)
```

(М.122)

$\mathrm{rows}(C_0) = 20211;$ $\mathrm{rows}(C_1) = 20221;$

$\mathrm{rows}(C_0) - \mathrm{rows}(C) = 20.$ $\mathrm{rows}(C_1) - \mathrm{rows}(C) = 30.$

```
ΣC0 := | for i ∈ 1 .. rows(C)
       |     ΣC0_i ← C_i + C0_i
       | ΣC0

ΣC1 := | for i ∈ 1 .. rows(C)
       |     ΣC1_i ← C_i + C1_i
       | ΣC1
```

(М.123)

$\mathrm{rows}(\Sigma C_0) = 20191.$ $\mathrm{rows}(\Sigma C_1) = 20191.$

Фрагменты (первые 70 отсчетов) результата сдвига сигнала **C** на $\delta_0$ и $\delta_1$ отсчетов изображены на рис. 5.90.

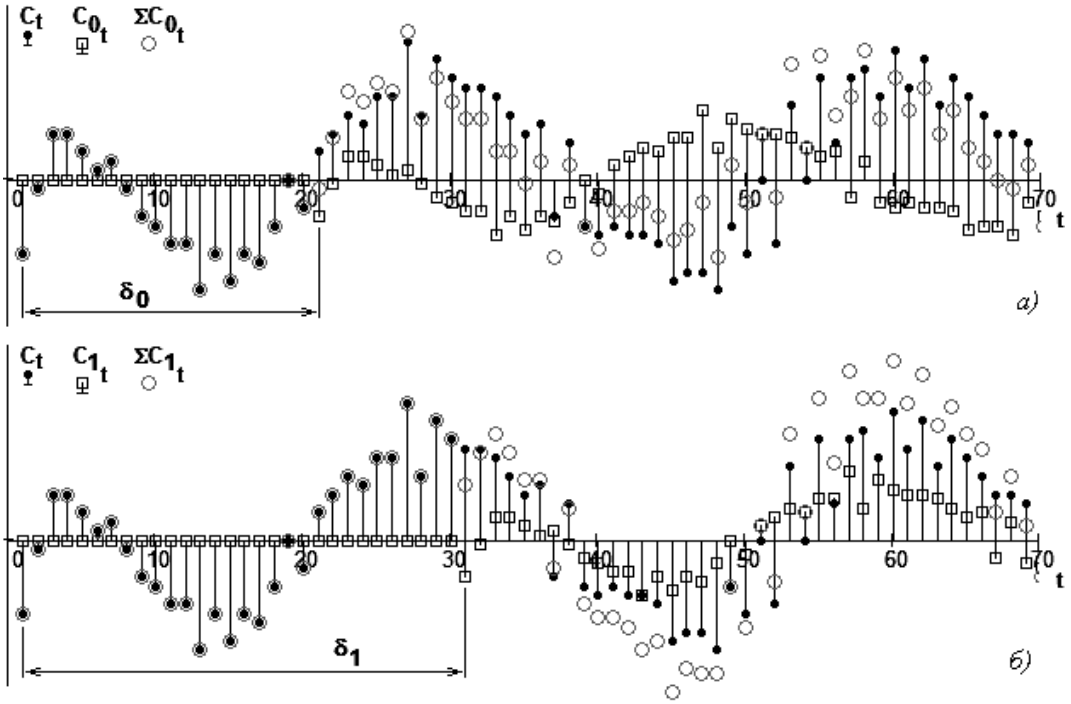



**Рис. 5.90. «Нулевой» (*а*) и «единичный» (*б*) эхо-сигналы массива C при параметре $\alpha = 0.5$**

**Шаг 3**

Для возможности скрытия более одного бита первичный сигнал **C** необходимо разделить на меньшие сегменты, каждый из которых будет рассматриваться как от-

дельный сигнал и в который путем эхо-отображения может быть встроен необходимый бит данных.

Вычислим количество отсчетов в одном сегменте, исходя из битовой длины сообщения $L_M$ (путем округления к ближайшему наименьшему целому с помощью функции **floor**(•)):

$$N_Б := \mathbf{floor}(\mathbf{rows}(C)/L_M), \quad N_Б = 280.$$

**Шаг 4**

Исходя из указанной при описании метода необходимости в наклонных фронтах импульсов смесительных сигналов (трапецеидальные импульсы), предварительно задаемся следующим:

- размах импульса: $U := 1$;
- длительность фронтов: $\tau := 20$ отсчетов;
- длительность импульса по уровню **U**: $T := N_Б - \tau$, $T = 260$ отсчетов;

Амплитуды отсчетов импульса формируем с помощью комплексного программного модуля (М.120).

$$\begin{aligned}
&fn := \left|\begin{array}{l}\text{for } n \in 1..\tau \\ \quad \left|\begin{array}{l} fn_n \leftarrow \dfrac{n-1}{\tau} \text{ if } \tau > 0 \\ fn_1 \leftarrow 0 \text{ if } \tau = 0 \end{array}\right. \\ fn\end{array}\right. \\
&u := \left|\begin{array}{l}\text{for } n \in (\tau+1)..(\tau+T) \\ \quad u_{n-\tau} \leftarrow 1 \\ u\end{array}\right. \\
&f3 := \left|\begin{array}{l}\text{for } n \in (\tau+T+1)..(2\cdot\tau+T) \\ \quad \left|\begin{array}{l} f3_{n-(\tau+T)} \leftarrow \dfrac{n-(2\cdot\tau+T)}{-\tau} \text{ if } \tau > 0 \\ f3_1 \leftarrow 0 \text{ if } \tau = 0 \end{array}\right. \\ f3\end{array}\right.
\end{aligned} \qquad \text{(М.124)}$$

Графическую интерпретацию вычисления модуля приведено на рис. 5.91.

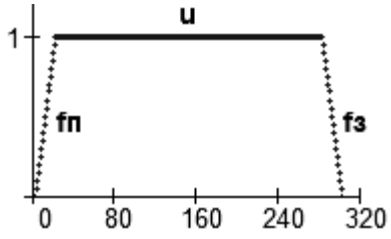



**Рис. 5.91. Трапецеидальный импульс, построенный по $2\cdot\tau + T$ отсчетам**

Формирование смесительных сигналов выполним, пользуясь программным модулем (М.125). При этом принимаем, что каждый бит сообщения кодируется половиной переднего фронта импульса, его единичным уровнем и половиной заднего фронта.

$$
\begin{aligned}
\mu_1 := \;& M_{vec} \leftarrow str2vec(M) \\
& M_{vec\_bin} \leftarrow D2B\left(M_{vec_1}\right) \\
& \text{for } j \in 2..N_M \\
& \quad M_{vec\_bin} \leftarrow stack\left(M_{vec\_bin}, D2B\left(M_{vec_j}\right)\right) \\
& \mu \leftarrow stack\left[0 \cdot submatrix\left[fs, \left(round\left(\frac{\tau}{2}\right)+1\right), \tau, 1, 1\right], 0 \cdot u\right] \text{ if } M_{vec\_bin_1} = 0 \\
& \mu \leftarrow stack\left[0 \cdot submatrix\left[fn, \left(round\left(\frac{\tau}{2}\right)+1\right), \tau, 1, 1\right]+1, u\right] \text{ if } M_{vec\_bin_1} = 1 \\
& \text{for } m \in 2..L_M \\
& \quad \mu \leftarrow stack(\mu, fs, 0 \cdot u) \text{ if } M_{vec\_bin_m} = 0 \wedge M_{vec\_bin_m} \neq M_{vec\_bin_{m-1}} \\
& \quad \mu \leftarrow stack(\mu, 0 \cdot fs, 0 \cdot u) \text{ if } M_{vec\_bin_m} = 0 \wedge M_{vec\_bin_m} = M_{vec\_bin_{m-1}} \\
& \quad \mu \leftarrow stack(\mu, fn, u) \text{ if } M_{vec\_bin_m} = 1 \wedge M_{vec\_bin_m} \neq M_{vec\_bin_{m-1}} \\
& \quad \mu \leftarrow stack[\mu, (0 \cdot fn + 1), u] \text{ if } M_{vec\_bin_m} = 1 \wedge M_{vec\_bin_m} = M_{vec\_bin_{m-1}} \\
& \text{while } rows(\mu) \le rows(C) \\
& \quad \mu \leftarrow stack(\mu, 0 \cdot fn, 0 \cdot u) \text{ if } M_{vec\_bin_{L_M}} = 0 \\
& \quad \mu \leftarrow stack[\mu, (0 \cdot fn + 1), u] \text{ if } M_{vec\_bin_{L_M}} = 1 \\
& \mu
\end{aligned}
\qquad \text{(M.125)}
$$

$$\mu_0 := \overrightarrow{(-1 \cdot \mu_1 + 1)}$$

После формирования смесительного сигнала для всех $L_M$ бит сообщения, данный сигнал дописывается нулевым или единичным уровнем, в зависимости от значения последнего ($L_M$-го) бита. Нулевой смесительный сигнал получается из единичного с учетом того, что их сумма должна равняться единице.

Смесительные сигналы для первой восьмерки бит сообщения представлены на рис. 5.92.

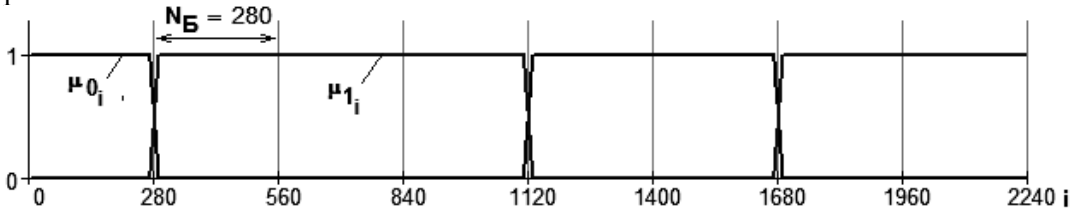



**Рис. 5.92. «Нулевой» ($\mu_0$) и «единичный» ($\mu_1$) смесительные сигналы**

**Шаг 5**

Непосредственно встраивание бит сообщения в аудиоконтейнер выполняет программный модуль (М.126).

В зависимости от значения текущего бита бинарного сообщения $M_{vec\_bin}$, проводится выделение сегмента заданной размерности ($N_Б$) из соответствующего эхо-сигнала ($\Sigma C_0$ или $\Sigma C_1$) и из смесительного сигнала ($\mu_0$ или $\mu_1$), которые в дальнейшем поэлементно перемножаются (для чего можно также использовать и операцию векторизации).

Полученные для каждого бита векторы **s'** формируют общий вектор заполненного контейнера **S**, в конец которого после встраивания последнего символа сооб-

щения дописываются элементы контейнера-оригинала, которые не были модифицированы. Очевидно, что количество элементов сформированного вектора **S** будет отвечать соответствующему показателю для вектора **C**, то есть **rows**(**S**) = 20191.

```
S := | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. N_M
     |     M_vec_bin ← stack(M_vec_bin , D2B(M_vec_j))
     | for m ∈ 1 .. L_M                                          (M.126)
     |   if M_vec_bin_m = 0
     |     Σ ← submatrix[ΣC_0 , [N_Б·(m − 1) + 1] , N_Б·m , 1 , 1]
     |     μ ← submatrix[μ_0 , [N_Б·(m − 1) + 1] , N_Б·m , 1 , 1]
     |     for n ∈ 1 .. N_Б
     |         S'_n ← Σ_n·μ_n
     |   if M_vec_bin_m = 1
     |     Σ ← submatrix[ΣC_1 , [N_Б·(m − 1) + 1] , N_Б·m , 1 , 1]
     |     μ ← submatrix[μ_1 , [N_Б·(m − 1) + 1] , N_Б·m , 1 , 1]
     |     for n ∈ 1 .. N_Б
     |         S'_n ← Σ_n·μ_n
     |   S ← S' if m = 1
     |   S ← stack(S , S') if m > 1
     | for i ∈ rows(S) + 1 .. rows(C) if rows(S) < rows(C)
     |     S_i ← C_i
     | S
```

Полученный вектор объединяем с немодифицированным вторым каналом, а результат объединения записываем в аудиофайл:

**WRITEWAV**("S_echo.wav", $f_д$, **Q**) := **augment**(**S**, $\mathbf{C_{total}}^{\langle 1 \rangle}$).

**Шаг 6**

Для извлечения скрытого сообщения предусматривается следующее: получателю известны размерность блоков, на которые разбивается контейнер (то есть $N^*_Б := N_Б$), а также значения нулевой и единичной задержек ($\delta^*_0 := \delta_0$ и $\delta^*_1 := \delta_1$).

Программный модуль извлечения данных — (М.127). В основу модуля положено вычисление автокорреляционной функции кепстра (5.61), однако для более надежного извлечения анализируется окрестность отсчетов $\delta^*_0$ и $\delta^*_1$.

На рис. 5.93 приведен результат вычисления АКФ кепстра для первого и восьмого бит сообщения ($M_{vec\_bin_1} = 0$ и $M_{vec\_bin_8} = 1$).

Результаты вычисления показателей звукового искажения контейнера при встраивании в него данных путем эхо-кодирования сведены в табл. 5.6.

```
M* := | m ← 1
      | while  N*Б·m < rows(S*)
      |        | s ← submatrix[S*, [N*Б·(m − 1) + 1], N*Б·m, 1, 1]
      |        | F ← CFFT(s)
      |        | L ← ln(F⃗ + 10^−20)^2
      |        | AC ← ICFFT(L)
      |        |          δ*0+2
      |        | "0" ←   Σ      |ACj|
      |        |         j = δ*0−2
      |        |          δ*1+2
      |        | "1" ←   Σ      |ACj|
      |        |         j = δ*1−2
      |        | B ← 1   if  "1" > "0"
      |        | B ← 0   if  "1" < "0"
      |        | B ← round(rnd(1))   if  "1" = "0"
      |        | M* bin m ← B
      |        | m ← m + 1
      | for  j ∈ 1 .. rows(M*bin) ÷ 8
      |        M*vecj ← B2D(submatrix(M*bin, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*vec)
```

(M.127)

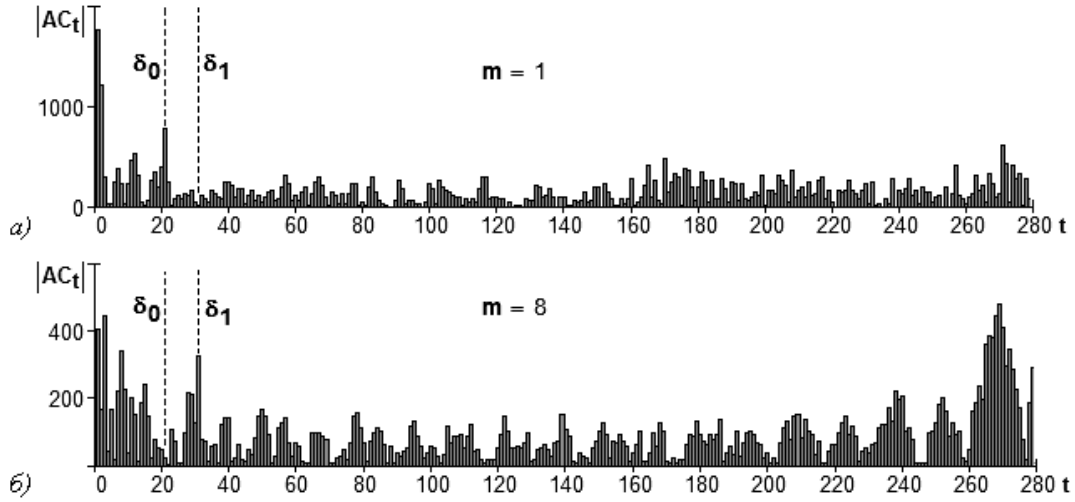



**Рис. 5.93. Пример вида АКФ кепстра сигнала, который содержит «нулевое» (*а*) и «единичное» (*б*) эхо-отображение**

**Таблица 5.6. Показатели звукового искажения в случае скрытия данных в аудиосреде**

| *Название показателя искажения* | *Оригинал* | *Методы скрытия данных в аудиосреде* | | | |
|---|---|---|---|---|---|
| | | *НЗБ (ПС интервал)* | *Фазовое кодирование* | *Расширение спектра* | *Эхо-кодирование* |
| Максимальная разность, ***MD*** | 0 | 1 | 1124 | 148 | 2963 |
| Средняя абсолютная разность, ***AD*** | 0 | $3.244 \cdot 10^{-3}$ | 15.298 | 14.321 | 572.093 |
| Нормированная средняя абсолютная разность, ***NAD*** | 0 | $2.625 \cdot 10^{-6}$ | 0.012 | 0.012 | 0.5 |
| Среднеквадратическая ошибка, ***MSE*** | 0 | 0.003 | $2.657 \cdot 10^{3}$ | 956.742 | $7.640 \cdot 10^{5}$ |
| Нормированная среднеквадратическая ошибка, ***NMSE*** | 0 | $9.402 \cdot 10^{-10}$ | $7.702 \cdot 10^{-4}$ | $2.773 \cdot 10^{-4}$ | 0.250 |
| $L^p$-норма, $p = 2$ | 0 | 0.057 | 51.551 | 30.931 | 874.01 |
| Отношение “сигнал/шум”, ***SNR*** | $\infty$ | $1.064 \cdot 10^{9}$ | $1.298 \cdot 10^{3}$ | $3.606 \cdot 10^{3}$ | 4.006 |
| Максимальное отношение “сигнал/шум”, ***PSNR*** | $\infty$ | $1.177 \cdot 10^{10}$ | $1.437 \cdot 10^{4}$ | $3.992 \cdot 10^{4}$ | 45.968 |
| Качество звучания, ***AF*** | 1 | $\approx 1$ | 0.999230 | 0.999723 | 0.750371 |
| Нормированная взаимная корреляция, ***NC*** | 1 | $\approx 1$ | 0.999615 | 0.999806 | 1.255078 |
| Качество корреляции, ***CQ*** | $2.792 \cdot 10^{3}$ | $2.792 \cdot 10^{3}$ | $2.790 \cdot 10^{3}$ | $2.791 \cdot 10^{3}$ | $3.355 \cdot 10^{3}$ |
| Структурное содержание, ***SC*** | 1 | $\approx 1$ | $\approx 1$ | 1.000110 | 0.568251 |
| Общее сигма-отношение “сигнал/шум”, ***GSSNR*** | $\infty$ | $1.152 \cdot 10^{14}$ | $1.51 \cdot 10^{20}$ | $3.279 \cdot 10^{8}$ | 9.377 |
| Сигма-отношение “сигнал/шум”, ***SSNR*** | $\infty$ | 140.6 | 201.8 | 85.2 | 9.721 |
| Отношение “сигма/ошибка”, ***SER*** | $\infty$ | $1.064 \cdot 10^{9}$ | $1.298 \cdot 10^{3}$ | $3.606 \cdot 10^{3}$ | 4.006 |
| Подобие гистограмм, ***HS*** | 0 | 240 | 13854 | 14708 | 15548 |

## 5.5. Скрытие данных в тексте

Для скрытия конфиденциальных сообщений в тексте (так называемая лингвистическая стеганография) используется или обычная избыточность письменной речи, или же форматы представления текста.

Наиболее сложным местом для скрытия данных по многим причинам является электронная (файловая) версия текста. В отличие от текстового файла его «жесткая» копия (например, бумажная) может быть обработана как высокоструктурированное изображение и поэтому является относительно легко поддающейся разнообразным методам скрытия, таким как незначительные изменения формата текстовых шаблонов, регулирование расстояния между определенными парами символов (кернинг), расстояния между строками и т.п. В значительной степени такая ситуация вызвана относительным дефицитом в текстовом файле избыточной информации, особенно в сравнении с графическими или, например, звуковыми файлами. В то время как в большинстве случаев в изображение и звук существует возможность внести незаметные глазу и неощутимые на слух модификации, даже дополнительная буква или знак пунктуации в тексте могут быть легко распознаны случайным читателем.

Скрытие данных в тексте требует поиска таких модификаций, которые были бы незаметными подавляющему большинству читателей. Авторы [14] рассматривают

три группы методов, которые получили наибольшее распространение при встраивании скрываемых данных в текст:

- *методы произвольного интервала*, которые осуществляют встраивание путем манипуляции с пробельными символами (свободным местом на печатной полосе);
- *синтаксические методы*, которые работают с пунктуацией;
- *семантические методы*, в основу алгоритмов которых положено манипулирование словами, зависимое от скрываемых бит данных.

### *5.5.1. Методы произвольного интервала*

Существует, по меньшей мере, две причины, по которым манипулирование свободным местом в определенных случаях показывает довольно неплохие результаты. Во-первых, изменение количества пробелов в конце текстовой строки не вызывает существенных изменений в значении фразы или предложения. Во-вторых, среднестатистический читатель вряд ли заметит незначительные модификации свободного места страницы текста.

В [14] предложено три метода, которые для скрытия данных используют свободное место в тексте. Указанные методы оперируют с интервалами между предложениями, пропусками в конце текстовых строк и интервалами между словами в тексте, выровненном по ширине.

#### 5.5.1.1. Метод изменения интервала между предложениями

Метод изменения интервала между предложениями позволяет встраивать в текст сообщение, имеющее двоичный формат, путем размещения одного или двух пробелов после каждого символа завершения предложения. В качестве символов окончания предложения могут служить, к примеру, точки в обычном тексте, точки с запятой для кода программ на языке C++ и т.п. При этом единичным пробелом может кодироваться бит «1», двойным — бит «0».

Кроме несомненной простоты, данный метод имеет и ряд недостатков.

Во-первых, он не эффективен, поскольку для встраивания незначительного количества бит требуется текст значительного объема. В частности, один бит, который возможно скрыть в одном предложении, эквивалентен скорости передачи данных, соответствующей приблизительно одному биту на 160 байт текстового контейнера, при условии, что в среднем предложение представляет собой две строки по 80 символов каждая.

Во-вторых, возможность скрытия весьма зависит от структуры текстового контейнера (некоторые тексты, как например, верлибры или свободные стихи характеризуются отсутствием стабильных согласованных или однозначных знаков завершения строки).

В-третьих, существуют текстовые редакторы, которые автоматически устанавливают после точки в конце предложения один-два пробела (так называемое автозавершение).

И, наконец, как отмечается в [14], непоследовательное и противоречивое использование свободных мест может оказаться достаточно заметным для читателя.

Приведем пример реализации данного метода в MathCAD.

**Шаг 1**

Пусть в качестве контейнера используется текст, фрагмент которого приведен на рис. 5.94 (**C** := **READBIN**("C.txt", "byte")), а скрываемое сообщение имеет следующее содержание: **M** := "Алгоритм".

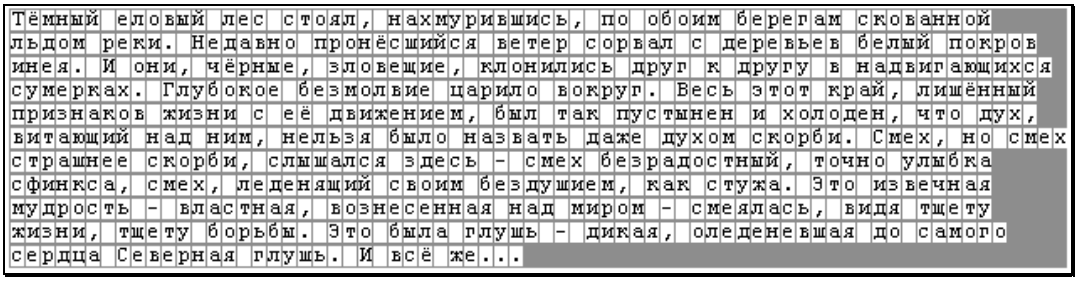

Тёмный еловый лес стоял, нахмурившись, по обоим берегам скованной
льдом реки. Недавно пронёсшийся ветер сорвал с деревьев белый покров
инея. И они, чёрные, зловещие, клонились друг к другу в надвигающихся
сумерках. Глубокое безмолвие царило вокруг. Весь этот край, лишённый
признаков жизни с её движением, был так пустынен и холоден, что дух,
витающий над ним, нельзя было назвать даже духом скорби. Смех, но смех
страшнее скорби, слышался здесь - смех безрадостный, точно улыбка
сфинкса, смех, леденящий своим бездушием, как стужа. Это извечная
мудрость - властная, вознесенная над миром - смеялась, видя тщету
жизни, тщету борьбы. Это была глушь - дикая, оледеневшая до самого
сердца Северная глушь. И всё же...

**Рис. 5.94. Фрагмент оригинала текста-контейнера, используемого для скрытия данных**[14]

**Шаг 2**

Переменной $\pi$ := ". " (точка + пробел) пометим фрагмент контейнера, который будет сигнализировать об окончании предложений. В ASCII-кодировке данная переменная отображается вектором $\mathbf{str2vec}(\pi)^T = (46\ \ 32)$.

Проверку достаточности количества предложений в тексте для скрытия заданного количества бит сообщения ($L_M := 8 \cdot \mathbf{strlen}(M)$, $L_M = 64$ бита) выполним с помощью программного модуля (М.128), который подсчитывает количество элементов $\pi$ в выбранном в качестве контейнера тексте.

$$
\begin{aligned}
N_\pi := \;& N_\pi \leftarrow 0 \\
& \textbf{for}\ \ i \in 1 .. \mathbf{rows}(C) \\
& \quad \Pi \leftarrow \mathbf{submatrix}(C, i, i+1, 1, 1) \ \ \textbf{if}\ \ i < \mathbf{rows}(C) \\
& \quad N_\pi \leftarrow N_\pi + 1 \ \ \textbf{if}\ \ \Pi = \mathbf{str2vec}(\pi) \\
& N_\pi
\end{aligned}
\qquad \text{(М.128)}
$$

Для представленного на рис. 5.94 фрагмента $N_\pi = 8$, что является достаточным для скрытия одного символа сообщения (8 бит).

**Шаг 3**

Стеганографическое скрытие рассматриваемым методом выполняет программный модуль (М.129). В начале модуля (М.129) строка символов **M** преобразовывается в вектор двоичных данных $\mathbf{M}_{\mathbf{vec\_bin}}$. Для каждого элемента вектора $\mathbf{M}_{\mathbf{vec\_bin}}$ проводится поиск элемента $\pi$ (конца предложения) в массиве исходного текста **C'**, представленном в ASCII-кодах. Параллельно формируется стеганограмма **S**, путем дописывания в ее конец **i**-го элемента массива **C'**.

В случае нахождения фрагмента массива **C'**, который отвечает вектору $\pi$, происходит сохранение дальнейшей части текста как немодифицированного (переопределение переменной **C'**) и в зависимости от значения текущего элемента $\mathbf{M}_{\mathbf{vec\_bin}}$ к стеганограмме **S** дописывается один или два пробела (ASCII-код которых равен 32).

Для предотвращения возможного возникновения неоднозначности в тех случаях, когда в оригинальном тексте предложения отделялись между собой точкой и произвольным количеством пробелов, в модуле (М.129) также предусмотрено удаление возможных лишних пробелов в начале немодифицированного фрагмента **C'**.

По окончании встраивания всех бит сообщения, к стеганограмме **S** приписывается фрагмент текста, оставшийся без изменений (**C'**).

[14] Использован отрывок из повести Дж. Лондона «Белый клык», пер. с англ. Н. Волжиной.

```
S := | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)                                          (M.129)
     | for j ∈ 2.. strlen(M)                                  if strlen(M) > 1
     |     M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | C' ← C
     | for μ ∈ 1.. 8·strlen(M)
     |     for i ∈ 1.. rows(C')
     |         S_rows(S)+1 ← C'_i
     |         Π ← submatrix(C', i, i + 1, 1, 1)   if i < rows(C')
     |         if M_vec_bin_μ = 1                                  if Π = str2vec(π)
     |             C' ← submatrix(C', i + 2, rows(C'), 1, 1)
     |             S_rows(S)+1 ← 32
     |             while C'_1 = 32
     |                     C' ← submatrix(C', 2, rows(C'), 1, 1)
     |             break
     |         if M_vec_bin_μ = 0                                  if Π = str2vec(π)
     |             C' ← submatrix(C', i + 2, rows(C'), 1, 1)
     |             S_rows(S)+1 ← 32
     |             S_rows(S)+1 ← 32
     |             while C'_1 = 32
     |                     C' ← submatrix(C', 2, rows(C'), 1, 1)
     |             break
     | stack(S, C')
```

Результат скрытия первого символа сообщения **M** (символа "A"), ASCII-код которого в двоичном представлении: **D2B**(**str2vec**("A"))$^{T}$ = ($0_{LSB}$ 0 0 0 0 0 1 $1_{MSB}$), представлен на рис. 5.95.

Тёмный еловый лес стоял, нахмурившись, по обоим берегам скованной
льдом реки. Недавно пронёсшийся ветер сорвал с деревьев белый покров
инея. И они, чёрные, зловещие, клонились друг к другу в надвигающихся
сумерках. Глубокое безмолвие царило вокруг. Весь этот край, лишённый
признаков жизни с её движением, был так пустынен и холоден, что дух,
витающий над ним, нельзя было назвать даже духом скорби. Смех, но смех
страшнее скорби, слышался здесь - смех безрадостный, точно улыбка
сфинкса, смех, леденящий своим бездушием, как стужа. Это извечная
мудрость - властная, вознесенная над миром - смеялась, видя тщету
жизни, тщету борьбы. Это была глушь - дикая, оледеневшая до самого
сердца Северная глушь. И всё же...

**Рис. 5.95. Фрагмент текстового контейнера, заполненного методом изменения интервала между предложениями**

**Шаг 4**

Извлечение скрытой информации проводится программным модулем (М.130) при **S*** := **S**, $\pi^*$ := $\pi$.

```
M* := | μ ← 1
      | for i ∈ 1 .. rows(S*)
      |     Π ← submatrix(S* , i , i + 1 , 1 , 1)  if  i < rows(S*)
      |     if  Π = str2vec(π*)
      |         M*bin_μ ← 0  if  S*_{i+1} = S*_{i+2} = 32
      |         M*bin_μ ← 1  if  S*_{i+1} ≠ S*_{i+2}
      |         μ ← μ + 1
      | for j ∈ 1 .. rows(M*bin) ÷ 8
      |     M*vec_j ← B2D(submatrix(M*bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M*vec)
```

(М.130)

Алгоритм действий следующий: выполняется поиск конца предложения (элемента $\pi^*$) в представленном в ASCII-кодах тексте **S***. В случае нахождения, анализируется количество пробелов после точки, на основании чего делается вывод о значении текущего извлекаемого бита сообщения. В конце модуля вектор двоичных данных преобразовывается в текстовую строку символов.

#### 5.5.1.2. Метод изменения количества пробелов в конце текстовых строк

Еще один метод использования свободных мест полосы текста для встраивания конфиденциальных данных заключается в добавлении пробелов в конец каждой текстовой строки. Количество добавляемых пробелов зависит от значения встраиваемого бита. Два пробела кодируют один бит на строку, четыре пробела — два бита, восемь — три и т.д. Такой подход позволяет существенно увеличить, по сравнению с предыдущим методом, количество информации, которую можно скрыть в тексте аналогичного объема.

Дополнительные преимущества указанного метода состоят в том, что он может быть применен к любому тексту. Изменения в формате последнего будут в достаточной степени незаметными, поскольку используемые при этом свободные места являются периферийными по отношению к основному тексту.

Недостатком данного (как, в конечном счете, и предыдущего) метода является то, что некоторые программы обработки текста могут непреднамеренно удалять дополнительно внесенные пробелы. Кроме того, характерным недостатком рассматриваемого метода является очевидная невозможность извлечения скрытых данных из бумажной копии текста (из-за невидимости пробелов).

Нами предложен следующий вариант реализации данного метода.

##### Шаг 1

Пусть незаполненный текстовый контейнер и скрываемое сообщение те же, что и в вышерассмотренном методе.

##### Шаг 2

Встраивание бит сообщения выполняем при помощи программного модуля (М.131).

```
S := | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2.. strlen(M)                                  if strlen(M) > 1
     |     M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | n ← 0
     | r ← 1
     | for i ∈ 1.. rows(C)
     |     | n ← n + 1   if C_i ≠ 10
     |     | if C_i = 10
     |     |     | L_r ← n − 1
     |     |     | n ← 0
     |     |     | r ← r + 1
     | C' ← C
     | r ← 1
     | μ ← 1
     | while μ ≤ 8·strlen(M)
     |       for i ∈ 1.. rows(C')
     |           | S_rows(S)+1 ← C'_i   if C'_i ≠ 13
     |           | if C'_i = 13
     |           |     | C' ← submatrix(C', i+2, rows(C'), 1, 1)   if i+2 ≤ rows(C')
     |           |     | κ ← 1
     |           |     | while max(L) − L_r ≥ κ
     |           |     |       | S_rows(S)+1 ← 32   if M_vec_bin_μ = 0
     |           |     |       | S_rows(S)+1 ← 160   if M_vec_bin_μ = 1
     |           |     |       | μ ← μ + 1
     |           |     |       | break   if μ > 8·strlen(M)
     |           |     |       | κ ← κ + 1
     |           |     | S_rows(S)+1 ← 13
     |           |     | S_rows(S)+1 ← 10
     |           |     | r ← r + 1
     |           |     | break
     | S ← stack(S, C')   if r < rows(L)
     | S
```

#

(M.131)

После получения вектора двоичного представления сообщения, которое подлежит скрытию, выполняется подсчет количества символов в каждой текстовой строке

**r** (напомним, что каждая строка текста завершается парой служебных символов возврата каретки — *CR* (ASCII-код 13), и переведения строки — *LF* (ASCII-код 10), которые в обычном режиме не выводятся на экран и на печать). Подсчитанное количество символов заносится в **r**-й элемент вектора **L**.

Изначально, как и в вышерассмотренном методе, массив **C'** представляет собой текст, который еще не претерпел изменений.

Стеганограмма формируется путем дописывания вслед за последним элементом представляющего ее вектора **S** значения **i**-го элемента массива **C'**, если указанный элемент не представляет собой символ *CR*. В противном случае совершается переопределение переменной **C'** как такой, которая содержит немодифицированную часть текста.

Количество бит **κ**, которое можно скрыть в строке, определяется разностью между количеством символов в наиболее длинной строке текста (максимальное значение вектора **L**) и количеством символов в текущей (**r**-й) строке. При этом, если скрывается бит «0», в стеганограмму дописывается значение 32 (обычный пробел), а если бит «1» — значение 160 (неразрывный пробел). После встраивания **κ** бит, строка завершается парой символов *CR*/*LF*.

После встраивания всех 8·**strlen**(**M**) бит сообщения цикл прерывается. Если строка, которая модифицируется последней, не является последней в тексте (общее количество строк текста можно определить как количество элементов вектора **L**), стеганограмма **S** дополняется немодифицированным фрагментом текста (**C'**).

Результат встраивания сообщения **M** в контейнер **C** изображен на рис. 5.96.


Тёмный еловый лес стоял, нахмурившись, по обоим берегам скованной
льдом реки. Недавно пронёсшийся ветер сорвал с деревьев белый покров
инея. И они, чёрные, зловещие, клонились друг к другу в надвигающихся
сумерках. Глубокое безмолвие царило вокруг. Весь этот край, лишённый
признаков жизни с её движением, был так пустынен и холоден, что дух,
витающий над ним, нельзя было назвать даже духом скорби. Смех, но смех
страшнее скорби, слышался здесь - смех безрадостный, точно улыбка
сфинкса, смех, леденящий своим бездушием, как стужа. Это извечная
мудрость - властная, вознесенная над миром - смеялась, видя тщету
жизни, тщету борьбы. Это была глушь - дикая, оледеневшая до самого
сердца Северная глушь. И всё же...

**Рис. 5.96. Фрагмент текстового контейнера, заполненного методом изменения количества пробелов в конце строк**

**Шаг 3**

Извлечение бит скрытых данных при **S*** := **S** выполняется при помощи программного модуля (М.132). При этом по всей длине вектора **S*** выполняется поиск символов *CR*. В случае нахождения проводится анализ символов, которые размещены до него, на предмет соответствия их кодов кодам обычного (32) и неразрывного (160) пробелов. По полученным результатам формируется вектор двоичных данных, который в дальнейшем конвертируется в строку символов **M***.

### 5.5.1.3. Метод изменения количества пробелов между словами выровненного по ширине текста

Данный метод позволяет скрывать данные в свободных местах текста, выровненного по ширине. При этом биты данных встраиваются путем управляемого выбора позиций, в которых будут размещены дополнительные пробелы. Один пробел между словами интерпретируется как «0». Два пробела — как «1». В среднем метод позволяет встраивать по несколько бит в одну строку.

```
M* := | μ ← 1
      | for i ∈ 1.. rows(S*)
      |     if S*_i = 13
      |         | j ← i
      |         | while S*_(j–1) = 32 ∨ S*_(j–1) = 160
      |         |         j ← j – 1
      |         | for m ∈ j.. i – 1                    if j ≠ i
      |         |     | M*bin_μ ← 0  if S*_m = 32
      |         |     | M*bin_μ ← 1  if S*_m = 160
      |         |     | μ ← μ + 1
      | for j ∈ 1.. rows(M*bin) ÷ 8
      |     M*vec_j ← B2D(submatrix(M*bin, 8·j – 7, 8·j, 1, 1))
      | vec2str(M*vec)
```

(M.132)

Из-за ограничений, которые накладываются выравниванием текста по ширине, не каждый пробел между словами может использоваться для встраивания данных. Для возможности принятия однозначного решения при определении принимающей стороной, какие же именно из пробелов между словами скрывают встроенную информацию, а какие являются частью оригинального текста, в [14] предложено использовать метод встраивания, аналогичный манчестерскому кодированию.

В результате упомянутого кодирования биты группируются попарно, причем последовательность «01» интерпретируется как «1», «10» — как «0», а пары «00» и «11» считаются пустыми. Например, извлеченное сообщение «1010000111» сводится к «001», тогда как сообщение «0011110011» представляет пустую строку.

Приведем вариант реализации данного метода в программе MathCAD.

#### Шаг 1

Примем, что пустой текстовый контейнер **C** и скрываемое сообщение **M** аналогичны использованным в двух вышерассмотренных методах.

#### Шаг 2

Встраивание сообщения в контейнер осуществляется с помощью программного модуля (М.133). Получение вектора двоичного представления скрываемого сообщения и подсчет количества символов в **r**-й текстовой строке выполняется аналогично тому, как это делалось в программном модуле (М.131) (обозначено символом «**#**»).

В процессе скрытия **μ**-го бита сообщения вычисляются индексы **i1** и **i2**, которые ограничивают **r**-ю текстовую строку в общем векторе **C**. В строку, которая имеет максимальное количество символов в своем составе (назовем ее длинной) — $\mathbf{L_r = max(L)}$, — данные не встраиваются. Во все остальные строки (назовем их короткими) — $\mathbf{max(L) - L_r > 0}$, — встраивание бит выполняется таким образом, чтобы добавление пробелов расширило текущую строку до размеров длинной строки.

Для выделенного из общего вектора **C** подмассива **c**, содержащего ASCII-коды символов короткой **r**-й строки, формируется массив **space**, размерность которого отвечает общему количеству пробелов **t** в данной строке, а каждые отдельные эле-

менты содержат информацию о порядковом номере соответствующих им пробелов среди всех символов строки. Например, при **r** = 1, для контейнера, изображенного на рис. 5.94,

$$\mathbf{space}^{T} = (7 \quad 14 \quad 18 \quad 25 \quad 39 \quad 42 \quad 48 \quad 56).$$

Из описания метода известно, что один бит данных встраивается в текстовый контейнер посредством модификации пары пробелов, которые заключают в себе одно слово предложения, через добавление одного пробела в начале или в конце слова (в зависимости от значения скрываемого бита). Следовательно, количество бит **Nb**, которое можно встроить в **r**-ю строку, определяется соотношением между количеством возможных пар пробелов и количеством символов, на которое короткая строка «отстает» от длинной строки.

Если $\mathbf{t}/2 < \mathbf{max}(\mathbf{L}) - \mathbf{L}_{\mathbf{r}}$, для встраивания бит можно использовать только $\mathbf{floor}(\mathbf{t}/2) - 1$ пар пробелов; остальные будут использоваться для удлинения строки до размеров длинной путем создания пустых пар. При этом для упрощения алгоритма принимается, что извлечение на определенном участке строки пустой пары пробелов указывает на то, что в последующей части строки встроенных данных не содержится.

Если $\mathbf{t}/2 \geq \mathbf{max}(\mathbf{L}) - \mathbf{L}_{\mathbf{r}}$, для встраивания бит используется количество пар пробелов, которое равняется $\mathbf{max}(\mathbf{L}) - \mathbf{L}_{\mathbf{r}}$, поскольку именно этого количества символов не хватает короткой строке, чтобы сравняться по длине с длинной.

```
S := # (M.131)
     Σ ← 0
     r ← 1
     μ ← 1
     while μ ≤ 8·strlen(M)
         break if r > rows(L)
         if r = 1
           | i1 ← 1
           | i2 ← L_r + 2
         if r ≥ 2
           | i1 ← i2 + 1
           | i2 ← i1 + L_r + 1
         if (max(L) − L_r) > 0
           | c ← submatrix(C, i1, i2, 1, 1)
           | Σ ← Σ + rows(c)
           | t ← 0
           | for i ∈ 1.. rows(c)
           |     if c_i = 32
           |         | space_{t+1} ← i
           |         | t ← t + 1
           | Nb ← floor(t ÷ 2) − 1 if t ÷ 2 < (max(L) − L_r)
           | Nb ← (max(L) − L_r) if t ÷ 2 ≥ (max(L) − L_r)
           | τ ← 1
```

(M.133)

продолжение модуля (M.133) на следующей странице

продолжение модуля (М.133)

```
while τ ≤ Nb
    break if μ > 8·strlen(M)
    sp2_τ ← space_{2·τ}
    seg_τ ← submatrix(c, 1, sp2_τ, 1, 1) if τ = 1
    seg_τ ← submatrix(c, sp2_{τ-1} + 1, sp2_τ, 1, 1) if τ > 1
    g ← 1
    for m ∈ 1.. rows(seg_τ)
        if (seg_τ)_m = 32
            ssp_g ← m
            g ← g + 1
    seg_new ← ins_vec2vec(seg_τ, 32, ssp_1 + 1) if M_vec_bin_μ = 1
    seg_new ← ins_vec2vec(seg_τ, 32, ssp_2 + 1) if M_vec_bin_μ = 0
    s_r ← seg_new if τ = 1
    s_r ← stack(s_r, seg_new) if τ > 1
    if τ = Nb
        seg_end ← submatrix(c, sp2_τ + 1, rows(c), 1, 1)
        if t ÷ 2 < (max(L) - L_r)
            g ← 1
            for m ∈ 1.. rows(seg_end)
                if seg_end_m = 32
                    ssp_g ← m
                    g ← g + 1
            seg_end_new ← ins_vec2vec(seg_end, 32, ssp_1 + 1)
            seg_end_new ← ins_vec2vec(seg_end_new, 32, ssp_2+1)
        seg_end_new ← seg_end if t ÷ 2 ≥ (max(L) - L_r)
        s_r ← stack(s_r, seg_end_new)
    μ ← μ + 1
    τ ← τ + 1
if L_r = max(L)
    c ← submatrix(C, i1, i2, 1, 1)
    Σ ← Σ + rows(c)
    s_r ← c
r ← r + 1
S ← s_1
for r ∈ 2.. rows(s)
    S ← stack(S, s_r)
S ← stack(S, submatrix(C, Σ + 1, rows(C), 1, 1)) if rows(s) < rows(L)
S
```

Так, например, для выбранного контейнера (рис. 5.94) длинной является шестая строка, которая содержит 70 символов. Первая строка является короткой: $\mathbf{L}_1 = 65$. Для нее количество возможных пар пробелов $\mathbf{t}/2 = 8/2 = 4$, что меньше чем $70 - 65 = 5$. Таким образом, $\mathbf{Nb} = 3$.

После определения количества пар пробелов, в которые могут быть встроены биты сообщения, из обрабатываемой строки выделяются сегменты $\mathbf{seg}_\tau$, которые, соответственно, содержат $\tau$-е пары пробелов. В нашем случае, для первой строки получаются следующие сегменты (для наглядности, изображены соответствующие ASCII-кодам символы):

$\mathbf{seg}_1 \rightarrow$ Тёмный еловый ;

$\mathbf{seg}_2 \rightarrow$ лес стоял, ;

$\mathbf{seg}_3 \rightarrow$ нахмурившись, по .

Для каждого сегмента формируется вектор **ssp**, состоящий из двух элементов, каждый из которых содержит индекс соответствующего ему пробела. Например, для отмеченных выше сегментов:

$$\mathbf{ssp} = \begin{pmatrix} 7 \\ 14 \end{pmatrix} \text{ для } \mathbf{seg}_1; \qquad \mathbf{ssp} = \begin{pmatrix} 4 \\ 11 \end{pmatrix} \text{ для } \mathbf{seg}_2; \qquad \mathbf{ssp} = \begin{pmatrix} 14 \\ 17 \end{pmatrix} \text{ для } \mathbf{seg}_3.$$

С помощью дополнительно определенной функции **ins_vec2vec(•)** (программный модуль (М.134)) формируются новые сегменты, в которых добавлен один пробел либо к первому, либо ко второму пробелу — в зависимости от значения текущего встраиваемого бита.

$$\begin{aligned}
&\mathbf{ins_vec2vec}(\mathbf{A}, \mathbf{B}, \rho) \equiv \\
&\quad \mathbf{A'} \leftarrow \mathbf{submatrix}(\mathbf{A}, 1, \rho - 1, 1, 1) \\
&\quad \text{if } \rho \le \mathbf{rows}(\mathbf{A}) \\
&\quad\quad \mathbf{A''} \leftarrow \mathbf{submatrix}(\mathbf{A}, \rho, \mathbf{rows}(\mathbf{A}), 1, 1) \\
&\quad\quad \mathbf{A'''} \leftarrow \mathbf{stack}(\mathbf{A'}, \mathbf{B}, \mathbf{A''}) \\
&\quad \mathbf{A'''} \leftarrow \mathbf{stack}(\mathbf{A'}, \mathbf{B}) \quad \text{if } \rho = \mathbf{rows}(\mathbf{A}) + 1 \\
&\quad \mathbf{A'''}
\end{aligned} \tag{М.134}$$

В общей сложности, функция **ins_vec2vec**(**A**, **B**, $\rho$) позволяет проводить встраивание вектора **B** в вектор **A**, начиная с позиции $\rho$.

В нашем применении в качестве вектора, в который происходит встраивание, выступает сегмент $\mathbf{seg}_\tau$. При этом ASCII-код пробела ($\mathbf{B} = 32$) встраивается после первого ($\rho = \mathbf{ssp}_1 + 1$), если $\mathbf{M}_{\mathbf{vec\_bin}} = 1$, или же после второго ($\rho = \mathbf{ssp}_2 + 1$), если $\mathbf{M}_{\mathbf{vec\_bin}} = 0$, пробела сегмента.

В частности, для приведенных выше сегментов получаются следующие модифицированные сегменты (напомним, что первым встраивается символ "A", код которого в двоичном эквиваленте — $\mathbf{D2B}(\mathbf{str2vec}(\text{"A"}))^T = (0_{LSB}\ 0\ 0\ 0\ 0\ 0\ 1\ 1_{MSB})$, а встраивание проводится, начиная с наименее значащих бит):

$\mathbf{seg}_{\mathbf{new}_1} \rightarrow$ Тёмный еловый  ;

$\mathbf{seg}_{\mathbf{new}_2} \rightarrow$ лес стоял,  ;

$\mathbf{seg}_{\mathbf{new}_3} \rightarrow$ нахмурившись, по  .

Полученные новые сегменты формируют **r**-ю строку стеганограммы ($\mathbf{s}_\mathbf{r}$).

После встраивания последнего (**Nb**-го) разрешенного в данной строке бита и формирования начала строки $\mathbf{s}_\mathbf{r}$, возможен случай, когда остается сегмент ориги-

нальной строки **c**, который не претерпел модификации — $\mathbf{seg_{end}}$. Для первой строки, например,

$$\mathbf{seg_{end}} \rightarrow \boxed{\text{обоим берегам скованной}}.$$

Если $\mathbf{t}/2 < \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, к первым двум пробелам данного завершающего сегмента прибавляются еще по пробелу, что позволит после присоединения этого сегмента к уже полученной части строки $\mathbf{s_r}$ получить строку, количество символов в которой будет равняться $\mathbf{max}(\mathbf{L})$. Если же выполняется условие $\mathbf{t}/2 \geq \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, завершающий сегмент не изменяется.

Для рассматриваемого выше примера, последний сегмент претерпевает модификацию, после которой приобретает следующий вид:

$$\mathbf{seg_{end\_new}} \rightarrow \boxed{\text{обоим  берегам  скованной}}.$$

Как уже было отмечено, вне зависимости от того, был или не был модифицирован завершающий сегмент, он дописывается в конец **r**-й строки стеганограммы.

На заключительном этапе полученные строки **s** стеганограммы объединяются в общий массив **S**.

Параллельно, в программном модуле (М.133) проводится подсчет количества символов обработанных строк **Σ**. Это позволяет определить индекс символа оригинального текста, после которого текст не претерпел изменений (например, по причине того, что скрываемое сообщение закончилось до того момента, как был достигнут конец текста), и дописать немодифицированный текст в конец полученной стеганограммы.

Фрагмент результата встраивания сообщения **M** в текстовый контейнер **C** приведен на рис. 5.97.

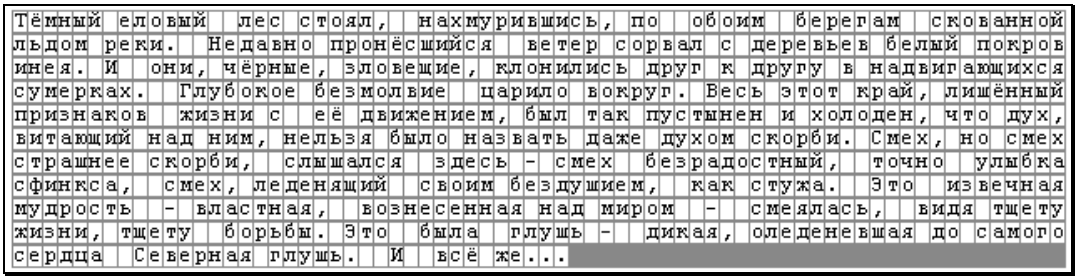



**Рис. 5.97. Фрагмент текстового контейнера, заполненного методом изменения количества пробелов между словами выровненного по ширине текста**

Нетрудно заметить, что в данном фрагменте скрыто 24 бита сообщения, что отвечает его первым трем символам ("Алг"). В частности, в 1-й строке скрыто биты 000, во 2-й — 00, в 3-й — 0, в 4-й — 11 (имеем символ "А"), в 5-й — 11, в 6-й — ни одного бита, в 7-й — 010, в 8-й — 111 (имеем символ "л"), в 9-й — 110, в 10-й — 0011, и, наконец, в 11-й — 1 (имеем символ "г").

#### Шаг 3

Извлечение из контейнера $\mathbf{S^*} := \mathbf{S}$ встроенных в него бит информации осуществляется с помощью программного модуля (М.135), представленного ниже.

В начале модуля выполняется подсчет количества символов в каждой из строк текста $\mathbf{S^*}$. Используя полученный при этом массива **L**, вычисляются индексы **i1** и **i2**, которые ограничивают **r**-ю текстовую строку в общем векторе $\mathbf{S^*}$.

```
M* := | n ← 0
      | r ← 1
      | for i ∈ 1 .. rows(S*)
      |     | n ← n + 1 if S*_i ≠ 10
      |     | if S*_i = 10
      |     |     | L_r ← n − 1
      |     |     | n ← 0
      |     |     | r ← r + 1
      | r ← 1
      | μ ← 1
      | while r ≤ rows(L)
      |     | if r = 1
      |     |     | i1 ← 1
      |     |     | i2 ← L_r + 2
      |     | if r ≥ 2
      |     |     | i1 ← i2 + 1
      |     |     | i2 ← i1 + L_r + 1
      |     | s ← submatrix(S* , i1 , i2 , 1 , 1)
      |     | t ← 0
      |     | for i ∈ 1 .. rows(s)
      |     |     if s_i = 32
      |     |         | space_{t+1} ← i
      |     |         | t ← t + 1
      |     | for j ∈ 1 .. t − 1
      |     |     | b_j ← 0 if (space_{j+1} − space_j) > 1
      |     |     | b_j ← 1 if (space_{j+1} − space_j) = 1
      |     | for τ ∈ 1 .. rows(b) / 3
      |     |     | tribit ← submatrix(b , 3·τ − 2 , 3·τ , 1 , 1)
      |     |     | if tribit^T = (0 1 0)
      |     |     |     | M*bin_μ ← 0
      |     |     |     | μ ← μ + 1
      |     |     | if tribit^T = (1 0 0)
      |     |     |     | M*bin_μ ← 1
      |     |     |     | μ ← μ + 1
      |     |     | break otherwise
      |     | r ← r + 1
      | for j ∈ 1 .. rows(M*bin) ÷ 8
      |     M*vec_j ← B2D(submatrix(M*bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M*vec)
```

(M.135)

Для выделенной из вектора **S*** строки-подмассива **s** формируется вектор **space**, размерность которого отвечает общему количеству пробелов в данной строке, а элементы содержат информацию о порядковых номерах соответствующих им пробелов среди всего множества символов строки.

На основе полученного вектора **space** формируется вектор **b**, значение каждого элемента которого зависит от результата вычисления разности между следующим и текущим значениями элементов вектора **space**. Если указанная разность равняется единице, элемент $\mathbf{b_j}$ принимает значение 1. Если же значение разности превышает единицу, то элементу $\mathbf{b_j}$ присваивается значение 0.

Из полученного вектора **b** выделяются подмассивы, состоящие из трех элементов (так называемые трибиты), на основе которых делается вывод о текущем значении встроенного бита: если трибит представляет собой последовательность (0 1 0), то извлекаемый бит — «0»; если же (1 0 0), извлекаемый бит — «1». Во всех других случаях принимается решение об отсутствии встроенных бит.

Сформированная таким образом двоичная последовательность преобразовывается в строку символьных данных, которая и возвращается переменной **M***.

Рассмотренные методы произвольного интервала эффективны, при условии, если текст представлен в формате ASCII. Как было уже отмечено выше, некоторые данные могут оказаться утраченными после распечатывания текста. Печатные документы выдвигают к скрытию данных такие требования, которые далеко выходят за возможности текстового файла при кодировании ASCII. При этом скрытие данных в «жестких» копиях текста может выполняться путем незначительных изменений расстояния между словами и отдельными буквами, изменением позиций базовых линий (линий, на которых лежат наиболее низкие элементы букв или знаков пунктуации строки), изменением форм символов и т.п.

### *5.5.2. Синтаксические и семантические методы*

Тот факт, что свободное место для встраивания выбирается произвольно, является одновременно как преимуществом, так и недостатком с точки зрения скрытости данных. Обычный читатель может и не заметить манипуляции с тестом, тогда как текстовый редактор способен автоматически изменить количество и размещение пробелов, таким образом разрушая скрытые данные.

Низкая стойкость к атакам, в свете возможного переформатирования документа, выступает одной из причин поиска других методов встраивания данных в текстовые контейнеры. Кроме этого, синтаксические и семантические методы вообще ни коим образом не используют свободные места в тексте, кардинально отличаясь от рассмотренных выше алгоритмов. Однако, все они могут использоваться одновременно, дублируя или же дополняя друг друга.

К *синтаксическим методам* текстовой стеганографии относятся методы изменения пунктуации и методы изменения структуры и стиля текста [14]. Существует немало случаев, когда правила пунктуации являются неоднозначными и несоблюдение их не влияет существенно на общее содержание текста. Так, например, фразы «красный, зеленый, синий» и «красный, зеленый и синий» эквивалентны друг другу. Тот факт, что выбор подобных форм может быть произвольным[15], и используется при построении стеганосистем на основе синтаксических методов. Периодическое

[15] Произвольным, разумеется, с позиций используемого в качестве контейнера текста, поскольку очевидно, что стеганосистема, построенная на основе видоизменения текста, известного широкому кругу лиц (например, классики), вряд ли может считаться надежной.

изменение форм при этом может быть поставлено в соответствие с двоичными данными. Например, появление в тексте формы перечисления с союзом «и» подразумевает под собой встроенный бит «1», в то время как отсутствие союза при перечислении будет говорить о том, что в данном случае встроен бит «0». Другим примером может служить использование сокращений и аббревиатур. Средняя скорость передачи данных такими методами составляет несколько бит на один килобайт текста [14].

Однако, в то время как письменный язык предоставляет достаточно возможностей для синтаксического скрытия данных, эти возможности исчезают в неповторимых классических произведениях. Кроме того, хотя некоторые из правил пунктуации и считаются неоднозначными, их противоречивое использование может стать объектом внимания для цензора. Также возможны случаи, когда изменение пунктуации приводит к снижению воспринимаемости текста или же к приобретению текстом диаметрально противоположного смысла. Поэтому синтаксические методы рекомендуется применять с осмотрительностью [14].

К синтаксическим методам также относятся методы изменения стиля и структуры текста без значительного изменения его смысловой нагрузки. Например, предложение «*Существует немало случаев, когда правила пунктуации являются неоднозначными*» можно сформулировать как «*Правила пунктуации являются неоднозначными во многих случаях*». Такие методы являются более незаметными для посторонних, по сравнению с методами изменения пунктуации, однако возможность их использования ограничена сложностью автоматизирования процесса стеганографического встраивания и извлечения бит сообщения.

*Семантические методы* подобны синтаксическим. Наряду с этим, вместо того чтобы встраивать двоичные данные, используя двусмысленность грамматической формы, семантические методы определяют два синонима, которые отвечают значениям скрываемых бит. К примеру, слово «*но*» может быть поставлено в соответствие к «0», а слово «*однако*» — к «1».

Для проведения скрытия с использованием семантических методов необходимо наличие таблицы синонимов. Кроме того, как отмечается в [14], если слову отвечает достаточно большое количество синонимов, возникает возможность одновременного кодирования большего количества бит. Скажем, выбор между синонимами «*секретный*», «*тайный*», «*скрытый*», «*конфиденциальный*», «*негласный*», «*неизвестный*», «*засекреченный*», «*закрытый*» дает возможность представить три бита данных за одно встраивание. Проблемы могут возникнуть, однако, когда желанию встроить бит информации препятствует нюанс значения слова.

## 5.6. Системные требования

Для эффективного применения предложенных программных комплексов рекомендуются следующие системные требования к рабочему месту.

*Минимальная конфигурация:* процессор Intel Pentium III / Athlon ~ 1 ГГц, оперативная память 256 Мб, видеоадаптер с 32 Мб, операционная система Microsoft Windows XP, универсальная математическая система MathCAD v.11, ~ 400 Мб свободного пространства на диске (преимущественно для установления системы MathCAD).

*Рекомендуемая конфигурация:* процессор Intel Pentium 4 / Athlon XP ~ 2.4 ГГц, оперативная память 512 Мб, видеоадаптер с 64 Мб, аудиоплата с периферией (для возможности оценки результатов скрытия данных в звуковых файлах, а также

записи звуков через микрофон), ОС Microsoft Windows XP, универсальная математическая система MathCAD v.11 и выше, ~ 400 Мб свободного пространства на диске.

Для формирования и обработки изображений используется любой графический редактор. В большинстве случаев достаточно наличия графического редактора *MS Paint*, который поставляется вместе с ОС MS Windows XP и позволяет работать с точечными изображениями формата JPG, GIF или BMP. С помощью MS Paint можно моделировать такие искажения как масштабирование изображения, его поворот, наклон, фрагментация, отображение по горизонтали или вертикали, инвертирование цветов. Для возможности моделирования атак наподобие добавления гауссового шума, фильтрации, экспозиции, изменения глубины цветов, масштабирования цветов изображения с использованием различных типов фильтров, передискретизации и т.п. рекомендуется также наличие одного из специализированных пакетов обработки изображения: *ACD FotoCanvas*, *Adobe Photoshop*, *Deformer*, *Focus Photoeditor*, *Gimp*, *JPEG Imager*, *PhotoStudio* или др.

Для работы с аудиостеганометодами в большинстве случаев достаточно возможностей программы *Звукозапись / Sound Recorder*, которая является прикладной программой из инструментария ОС MS Windows XP. Указанная программа позволяет осуществлять запись, смешивание, воспроизведение и редактирование звукозаписей. Кроме того, она позволяет выполнять следующие операции над импортированным аудиофайлом: удаление части звукозаписи, изменение скорости и громкости воспроизведения, изменение направления воспроизведения, изменение и конвертация типов звукозаписей, добавление эха. Для моделирования искажений можно использовать и такие программы как *Nero Wave Editor*, *AudioEdit Deluxe*, *SoundForge*, *GoldWave*, *AV Voice Changer Software*, *Audio Editor Gold* и др.

При исследовании методов лингвистической стеганографии вполне достаточно возможностей таких текстовых редакторов как *Notebook*, *WordPad*, *MS Office Word* и т.п.

## 5.7. Выводы

В данной главе были подробно рассмотрены известные на сегодня стеганографические методы скрытия данных в неподвижных изображениях, в аудиосигналах и текстовых файлах. Приведены примеры программных комплексов, которые демонстрируют принципы, заложенные в основу более чем двадцати методов стеганографического скрытия информации в пространственной, временной и частотной областях используемого контейнера. Разработка комплексов проведена с использованием популярной математической системы MathCAD.

Все этапы скрытия проиллюстрированы результатами, полученными авторами во время моделирования. Приведены последствия возможных атак на стеганосообщение и даны рекомендации по защите от них. Также рассчитаны основные показатели визуального и звукового искажений, что позволяет провести предварительный анализ оптимальности выбранного формата контейнера для скрытия определенного типа данных.

# Заключение

Как показывает практика, за последние несколько лет актуальность проблемы информационной безопасности неуклонно возрастала, постоянно стимулируя при этом поиск новых методов защиты информации. Увеличение спроса на эффективные системы защиты информации является следствием не только всем понятной заинтересованности государственных структур, которым, как правильно отмечено в [5], необходимо теперь противостоять как разведкам других стран, так и внутренним противникам — адептам мирового терроризма или хакерства, но и очевидного желания руководителей больших и малых фирм и, в конечном счете, обычных граждан уберечь имеющиеся у них конфиденциальные данные от утечки и разглашения. Анализ существующей тенденции дает возможность утверждать, что и в следующие ближайшие годы интерес к внедрению и развитию методов эффективной защиты информации будет только возрастать, чему, собственно, будет содействовать и то стремительное развитие информационных технологий, которое мы наблюдаем сегодня.

Сделать заметный вклад в общее дело сохранения государственной тайны, уверенности фирмы в честной игре конкурентов, а гражданина — в действенности такого призабытого на сегодняшний день понятия как свобода личности, наряду с другими призваны и стеганографические методы защиты информации, а именно, методы компьютерной стеганографии, теоретические и практические основы которой изложены в предложенном учебнике. Полученные при этом результаты заключаются в следующем:

- проведен анализ специализированных литературных источников и ресурсов сети Internet относительно перспективных направлений, по которым возможно использование стеганографии как инструмента защиты информации в автоматизированных системах обработки данных, что позволило в отличие от распространенной в преобладающем большинстве работ схемы заключенных перейти к схеме стеганографической системы, отвечающей основным принципам теории связи;
- на основании исследования известных публикаций отечественных и зарубежных авторов выполнено системное изложение проблем надежности и стойкости произвольной стеганографической системы по отношению к видам совершаемых на нее атак, а также оценки пропускной способности канала скрытого обмена данными, каким, по сути, является стеганосистема. Выделены как общие, так и характерные только для стеганографических систем виды атак, возможность существования которых необходимо учитывать при организации канала скрытой связи. Приведены результаты существующих информационно-теоретических исследований проблемы информационного скрытия в случае активного противодействия нарушителя. В результате этого было заложено обоснованную теоретическую базу для разработки компьютерных систем стеганографического скрытия конфиденциальной информации;
- изложены принципы, положенные в основу большинства известных на сегодня стеганографических методов, направленных на скрытие конфиденциальных данных в компьютерных файлах графического, звукового и текстового форматов (в общей сложности рассмотрено более двадцати методов);
- для указанных методов проанализированы особенности соответствующих аппаратов человека (зрительного и слухового), сделан акцент на характерные нюансы, позволяющие воспользоваться существующими ограниченностями ЗСЧ и ССЧ в стеганографических целях. Сформулированы рекомендации относитель-

но возможных способов встраивания бит скрываемого сообщения в контейнер с целью повышения уровня скрытости встроенных данных при применении известных алгоритмов стеганоанализа;

- для демонстрации принципов, заложенных в методы стеганографической защиты информации, приведены примеры компьютерных стеганосистем, разработанных на их основе с использованием мощной математической системы MathCAD. Все этапы скрытия сопровождены соответствующими программными модулями, общее количество которых превысило 130. Полученные в процессе моделирования результаты проиллюстрированы значительным количеством (около сотни) графического материала. Проведено вычисление показателей искажения медиаконтейнеров при встраивании в них бит скрываемых данных, что позволяет проводить предварительный анализ оптимальности выбора того или иного стеганометода или же формата контейнера;
- разработанные системы позволяют проводить стеганографическое скрытие файлов практически любого формата в файлах растрового изображения форматов BMP, JPG, GIF и др., в файлах ИКМ аудиосигнала формата WAV, а также в текстовых файлах. Основные требования, которые при этом должны выполняться, заключаются в следующем: выбор файла-контейнера надлежащего объема и, конце концов, аппаратные возможности используемой вычислительной системы.

Сравнение стеганографии с технологиями криптографической защиты информации позволяет предположить их взаимную интеграцию уже в течение ближайшего будущего. При этом возникнет возможность избавиться от уязвимых сторон известных на сегодня методов защиты информации и разработать более эффективные и оптимальные с позиций вычислительной сложности и стойкости к взлому новые методы обеспечения информационной безопасности.

В заключение, нельзя не согласиться с авторами [5] относительно перспективного будущего цифровой и, в частности, компьютерной стеганографии, поскольку со времени ее зарождения прошло всего лишь 10 лет, а как показывает нам история, время, за которое технология с момента возникновения достигает стадии распространенного промышленного использования, обычно составляет около 15...20 лет.

## *Приложение А*

## Встроенные операторы MathCAD

В приведенном ниже перечне операторов используются следующие условные обозначения:

**A** и **B** – массивы (векторы или матрицы);

**u** и **v** – векторы с действительными или комплексными элементами;

**M** – квадратная матрица;

**z** и **w** – действительные или комплексные числа;

**x** и **y** – действительные числа;

**m** и **n** – целые числа;

**i** – диапазон переменных (дискретный аргумент);

**t** – любое имя переменной;

**f** – любая функция;

**X** и **Y** – переменные или выражения любого типа.

На их место могут подставляться объекты соответствующего типа (переменные, векторы, матрицы и т.п.) с любыми другими именами и выражения с результатом их вычисления соответствующего типа.

| *Оператор* | *Обозначение* | *Введение* | *Назначение оператора (операции)* |
|---|---|---|---|
| Присвоение значения полуглобальной переменной (элементу массива, столбцу матрицы), определение функции пользователя | ■ := ■<br>**z** := **x** + **y**<br>$\mathbf{B}_{n,m}$ := 8<br>$\mathbf{A}^{<n>}$ := **v**<br>**f**(**x**, [**y**,...]) := **x**+7 | : | Присваивает значение переменной (элементу массива), определяет функцию пользователя, видимую правее и ниже данного оператора |
| Присвоение значения глобальной переменной (элементу массива, столбцу матрицы), определение функции пользователя | ■ ≡ ■<br>**z** ≡ **x** + **y**<br>$\mathbf{B}_{n,m}$ ≡ 8<br>$\mathbf{A}^{<n>}$ ≡ **v**<br>**f**(**x**, [**y**,...]) ≡ **x**+7 | ~ | Присваивает значение переменной (элементу массива), определяет функцию пользователя, видимую во всем документе MathCAD |
| Присвоение значения локальной переменной (элементу массива, столбцу матрицы) | ■ ← ■<br>**z** ← **x** + **y**<br>$\mathbf{B}_{n,m}$ ← 8<br>$\mathbf{A}^{<n>}$ ←**v** | { | Присваивает значение локальной переменной (элементу массива) в пределах программного модуля |
| Вычисление числового значения | ■ = ■·[■]<br>**z** = 1980<br>$\mathbf{B}_{n,m}$ = 8<br>$\mathbf{v}_n$ = 1·s<br>**f**(**x**) = 74 | = | Вычисляет и выводит на экран во второй операнд числовое значение переменной, выражения, функции, записанной в первом операнде. Можно задавать размерность (третий операнд) |
| Круглые скобки | (**X**) | ' | Изменение приоритета выполнения операций, группа операторов |

| *Оператор* | *Обозначение* | *Введение* | *Назначение оператора (операции)* |
|---|---|---|---|
| Нижний индекс | $\mathbf{A_n}$ | `[` | Задание индексированной переменной |
| Верхний индекс | $\mathbf{A^{<n>}}$ | `[Ctrl]6` | Выбор **n**-го столбца из массива **A** |
| Векторизация | $\overrightarrow{\mathbf{f(A)}}$ | `[Ctrl]-` | Выполнение заданной операции **f** для всех элементов массива **A** |
| Факториал | $\mathbf{n}!$ | `!` | Вычисление факториала для целого положительного числа **n** |
| Сопряженное комплексное число | $\overline{\mathbf{z}}$ | `"` | Вычисление сопряженного комплексного числа (инвертированный сигнал мнимой части **z**) |
| Транспонирование | $\mathbf{A}^{\mathsf{T}}$ | `[Ctrl]1` | Транспонирование массива **A** |
| Возведение в степень | $\mathbf{z^w}$ | `^` | Возведение **z** в степень **w** |
| Степень матрицы, инверсная матрица | $\mathbf{M^n}$ | `^` | Возведение квадратной матрицы **M** в степень **n** (при **n** = –1 инверсия матрицы) |
| Отрицание | $-\mathbf{X}$ | `-` | Произведение **X** на –1 |
| Сумма элементов вектора | $\Sigma\mathbf{v}$ | `[Ctrl]4` | Вычисление суммы элементов вектора **v** (возвращается скалярное значение) |
| Квадратный корень | $\sqrt{\mathbf{w}}$ | `\` | Вычисление квадратного корня из **w** |
| Корень **n**-й степени | $\sqrt[\mathbf{n}]{\mathbf{w}}$ | `[Ctrl]\` | Вычисление корня степени **n** из **w** |
| Модуль комплексного числа | $\lvert\mathbf{z}\rvert$ | `\|` | Вычисление модуля $\sqrt{\mathbf{Re}(\mathbf{z})^2+\mathbf{Im}(\mathbf{z})^2}$ комплексного числа **z** |
| Абсолютная величина вектора (евклидова норма или размерность вектора) | $\lvert\mathbf{v}\rvert$ | `\|` | Вычисление $\sqrt{\mathbf{v}\cdot\mathbf{v}}$, если все элементы **v** являются действительными, и $\sqrt{\mathbf{v}\cdot\overline{\mathbf{v}}}$, если элементы **v** комплексные |
| Детерминант матрицы **M** | $\lvert\mathbf{M}\rvert$ | `\|` | Возвращает определитель (детерминант) квадратной матрицы **M** |
| Деление | $\frac{\mathbf{X}}{\mathbf{z}}$ или $\mathbf{X}\div\mathbf{z}$ | `/` | Деление выражения **X** на скаляр **z**, который не равняется 0 (если **X** является массивом, то на **z** делится каждый элемент массива) |
| Умножение | $\mathbf{X}\cdot\mathbf{Y}$ | `*` | Вычисление произведения **X** на **Y**, если **X** и **Y** являются скалярами. Умножение каждого элемента **Y** на **X**, если **Y** является массивом, а **X** — скаляром. Вычисление скалярного произведения, если **X** и **Y** — векторы одинаковой размерности. Умножение матриц, если **X** и **Y** — подобные матрицы |

| *Оператор* | *Обозначение* | *Введение* | *Назначение оператора (операции)* |
|---|---|---|---|
| Кросс-произведение | $\mathbf{u}\cdot\mathbf{v}$ | **[Ctrl]8** | Вычисление векторного произведения векторов **u** и **v** |
| Суммирование для конечного ряда | $\sum_{i=m}^{n} X$ | **[Ctrl] [Shift] 4** | Вычисление суммы членов **X** для **i** = **m**, **m**+1,..., **n** , причем **X** может быть любым выражением |
| Произведение для конечного ряда | $\prod_{i=m}^{n} X$ | **[Ctrl] [Shift] 3** | Перемножение элементов **X** для **i = m**, **m**+1,..., **n**, причем **X** может быть любым выражением |
| Суммирование для бесконечного ряда | $\sum_{i} X$ | **$** | Вычисление суммы бесконечного количества членов **X**, причем **X** может быть любым выражением |
| Произведение для бесконечного ряда | $\prod_{i} X$ | **#** | Перемножение бесконечного количества членов **X**, причем **X** может быть любым выражением |
| Граница функции в заданной точке | $\lim_{x \to a} f(x)$ | **[Ctrl]L** | Вычисление границы функции **f**(**x**) при **x**, стремящемся к **a** (только в символьном виде) |
| Граница функции слева от заданной точки | $\lim_{x \to a^-} f(x)$ | **[Ctrl]B** | Вычисление границы функции **f**(**x**) при **x**, стремящемся к **a** слева (только в символьном виде) |
| Граница функции справа от заданной точки | $\lim_{x \to a^+} f(x)$ | **[Ctrl]A** | Вычисление границы функции **f**(**x**) при **x**, стремящемся к **a** справа (только в символьном виде) |
| Определенный интеграл | $\int_{a}^{b} f(t)\,dt$ | **&** | Вычисление определенного интеграла от подынтегральной функции **f**(**t**) с пределами интегрирования **a** и **b** |
| Неопределенный интеграл | $\int f(t)\,dt$ | **[Ctrl]I** | Вычисление в символьном виде неопределенного интеграла от подынтегральной функции **f**(**t**) |
| Производная заданной функции по переменной **t** | $\frac{d}{dt} f(t)$ | **?** | Вычисление первой производной функции **f**(**t**) по переменной **t** |
| **n**-я производная заданной функции по переменной **t** | $\frac{d^n}{dt^n} f(t)$ | **[Ctrl]?** | Вычисление **n**-й производной функции **f**(**t**) по переменной **t** |
| Сложение | **X+Y** | **+** | Скалярное сложение, если **X** и **Y** скаляры. Сложение элементов, если **X** и **Y** — массивы одинаковой размерности. Если **X** массив, а **Y** — скаляр, сложение каждого элемента **X** с **Y** |
| Вычитание | **X–Y** | **–** | Выполняет вычитание скаляров, векторов или матриц **X** и **Y** |

| *Оператор* | *Обозначение* | *Введение* | *Назначение оператора (операции)* |
|---|---|---|---|
| Перенесение на другую строку | **X** ...<br>**+Y** | **[Ctrl]**<br>↵ | Перенесение части выражения на следующую строку. Линия разрыва имеет редакционное значение |
| Больше, чем | **x > y**<br>**"T1" > "T2"** | **>** | Возвращает 1, если **x > y**, иначе — 0. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и **T2** — возвращает 1, если ASCII-кодирование переменной **T1** больше, чем у переменной **T2** |
| Меньше, чем | **x < y**<br>**"T1" < "T2"** | **<** | Возвращает 1, если **x < y**, иначе — 0. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и**T2** — возвращает 1, если ASCII-кодирование переменной **T1** меньше, чем у переменной **T2** |
| Больше, чем или равняется | **x** ≥ **y**<br>**"T1"** ≥ **"T2"** | **[Ctrl]0** | Возвращает 1, если **x** ≥ **y**, 0 — в ином случае. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и **T2** — возвращает 1, если ASCII-кодирование **T1** ≥ **T2** |
| Меньше, чем или равняется | **x** ≤ **y**<br>**"T1"** ≤ **"T2"** | **[Ctrl]9** | Возвращает 1, если **x** ≤ **y**, 0 — в другом случае. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и **T2** — возвращает 1, если ASCII-кодирование **T1** ≤ **T2** |
| Равняется | **x = y**<br>**"T1" = "T2"** | **[Ctrl]=** | Возвращает 1, если **x = y**, 0 – в другом случае. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и **T2** – возвращает 1, если ASCII-кодирование **T1 = T2** |
| Не равняется | **x** ≠ **y**<br>**"T1"** ≠ **"T2"** | **[Ctrl]3** | Возвращает 1, если **x** ≠ **y**, 0 — в ином случае. **x** и **y** должны быть действительными скалярными переменными. Для текстовых переменных **T1** и **T2** — возвращает 1, если ASCII-кодирование **T1** ≠ **T2** |

## *Приложение В*

## Основные встроенные функции и директивы MathCAD

В приведенных ниже функциях MathCAD используются такие условные обозначения:

| | |
|---|---|
| **A** и **B** | – массивы (векторы или матрицы); |
| **M** и **N** | – квадратные матрицы; |
| **X** и **Y** | – переменные или выражения любого типа; |
| **F** | – вектор-функция; |
| **f** | – функция-скаляр; |
| **S** | – строчная переменная или константа |
| **x** и **y** | – действительные числа; |
| **z** и **w** | – действительные или комплексные числа; |
| **m**, **n**, **i**, **j**, **k** | – целые числа; |
| **u** и **v**[...] | – векторы; |
| *file* | – имя файла или файловая переменная, присоединенная к имени файла. |

Все углы в тригонометрических функциях выражены в радианах. Многозначные функции и функции с комплексным аргументом всегда возвращают главное значение. Имена приведенных функций нечувствительны к шрифту, однако чувствительны к регистру — их следует вводить в точности, как они приведены. Все функции возвращают указанное для них значение.

### *Функции для работы с комплексными числами*

**arg**(**z**) — возвращает аргумент комплексного числа **z** между $-\pi$ и $\pi$.

**csgn**(**z**) — возвращает комплексный знак числа: 0, если **z** = 0; 1, если **Re**(**z**) > 0 или (**Re**(**z**) = 0 и **Im**(**z**) > 0); –1 в другом случае.

**Im**(**z**) — возвращает мнимую часть комплексного аргумента **z**.

**Re**(**z**) — возвращает действительную часть комплексного аргумента **z**.

**signum**(**z**) — возвращает комплексный знак числа: 1, если **z** = 0; **z**/|**z**| в другом случае.

### *Кусочно-непрерывные функции*

**if**(**cond**, **f1**, **f2**) — условное выражение, возвращающее выражение **f1**, если логическое условие **cond** является правдивым, и выражение **f2** в других случаях.

$\delta$(**m**, **n**) — функция Кронекера: 1, если **m** = **n**; 0, если **m** $\neq$ **n**.

**Ф**(**x**) — единичная ступенчатая функция (функция Хевисайда): 1 при **x** $\geq$ 0; 0 при **x** < 0.

**sign**(**x**) — возвращает знак числа **x**: 0, если **x** = 0; 1, если **x** > 0; –1, если **x** < 0.

***Статистические функции и функции анализа данных***

**corr**(**A**, **B**) — коэффициент корреляции элементов массивов **A** и **B** по Пирсону. Вычисляется как $\frac{\mathbf{cvar}(\mathbf{A},\mathbf{B})}{\mathbf{stdev}(\mathbf{A})\cdot\mathbf{stdev}(\mathbf{B})}$.

**cvar**(**A**, **B**) — ковариация элементов массивов **A** и **B**, которые имеют размерность **m**×**n**, $\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left[\mathbf{A}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{A})\right]\cdot\overline{\left[\mathbf{B}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{B})\right]}$.

**gmean**(**A**,**B**,**C**,...) — возвращает среднее геометрическое для аргументов **A**, **B**, **C**, ... размерностью **m**×**n** по выражению $\sqrt[\mathbf{m}\cdot\mathbf{n}]{\prod_{\mathbf{i}=0}^{\mathbf{m}-1}\prod_{\mathbf{j}=0}^{\mathbf{n}-1}\frac{1}{\mathbf{M}_{\mathbf{i},\mathbf{j}}}}$, где **M** – массив, образованный аргументами **A**, **B**, **C**, ... .

**hist**(**int**, **d**) — возвращает вектор значения частот, с которыми величины, содержащиеся в векторе **d**, попадают в интервалы, представленные границами, заданными в векторе **int**.

**histogram**(**n**, **d**) — возвращает матрицу, которая имеет две колонки: первая содержит середины **n** подынтервалов диапазона равной длины, вторая является идентичной вычислению функции **hist**(**int**, **d**).

**hmean**(**A**,**B**,**C**,...) — возвращает среднее гармоничное для аргументов **A**, **B**, **C**, ... по формуле $\left(\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\frac{1}{\mathbf{M}_{\mathbf{i},\mathbf{j}}}\right)^{-1}$.

**kurt**(**A**, **B**, **C**, ...) — возвращает эксцесс аргументов **A**, **B**, **C**, ... по формуле $\left[\frac{\mathbf{m}\cdot\mathbf{n}\cdot(\mathbf{m}\cdot\mathbf{n}+1)}{(\mathbf{m}\cdot\mathbf{n}-1)\cdot(\mathbf{m}\cdot\mathbf{n}-2)\cdot(\mathbf{m}\cdot\mathbf{n}-3)}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left(\frac{\mathbf{M}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{M})}{\mathbf{Stdev}(\mathbf{M})}\right)^4\right]-\frac{3\cdot(\mathbf{m}\cdot\mathbf{n}-1)^2}{(\mathbf{m}\cdot\mathbf{n}-2)\cdot(\mathbf{m}\cdot\mathbf{n}-3)}$, причем массив **M**, образованный на основе аргументов **A**, **B**, **C**, ..., должен иметь не менее 4-х элементов, а стандартное отклонение этих элементов $\sigma\neq 0$.

**mean**(**A**, **B**, **C**, ...) — возвращает среднее арифметическое для аргументов **A**, **B**, **C**, ...: $\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\mathbf{M}_{\mathbf{i},\mathbf{j}}$.

**median**(**A**,**B**,**C**,...) — возвращает медиану для аргументов **A**, **B**, **C**, ... .

**mode**(**A**, **B**, **C**, ...) — возвращает значение элемента аргументов **A**, **B**, **C**, ..., которое встречается наиболее часто.

**skew**(**A**, **B**, **C**, ...) — возвращает коэффициент асимметрии значений аргументов **A**, **B**, **C**, ..., вычисленный как $\frac{\mathbf{m}\cdot\mathbf{n}}{(\mathbf{m}\cdot\mathbf{n}-1)\cdot(\mathbf{m}\cdot\mathbf{n}-2)}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left(\frac{\mathbf{M}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{M})}{\mathbf{Stdev}(\mathbf{M})}\right)^3$.

**stderr**(**u**, **v**) — возвращает среднеквадратическую ошибку, которая отвечает простой линейной регрессии для точек, описанных векторами

**u** и **v** (для каждого вектора количество элементов $\mathbf{n} \ge 3$). При этом определяется, насколько близко результаты обработки данных размещены к линии регрессии,

$$\sqrt{\frac{1}{\mathbf{n}-2}\cdot\sum_{\mathbf{i}}[\mathbf{v_i}-\mathbf{intercept}(\mathbf{u},\mathbf{v})+\mathbf{slope}(\mathbf{u},\mathbf{v})\cdot\mathbf{u_i}]^2}\ .$$

**Stdev**(**A**, **B**, **C**, ...) — возвращает значение выборочного стандартного отклонения значений аргументов, как корень квадратный из дисперсии **Var**(**A**, **B**, **C**, ...).

**stdev**(**A**, **B**, **C**, ...) — возвращает значение стандартного отклонения генеральной совокупности значений аргументов, как корень квадратный с дисперсии **var**(**A**, **B**, **C**, ...).

**Var**(**A**, **B**, **C**, ...) — возвращает значение выборочной дисперсии значений аргументов, вычисленное как $\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}(\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M}))^2 \Big/ (\mathbf{m}\cdot\mathbf{n}-1)$. Деление квадратов отклонения на "объем выборки минус один" позволяет получить лучшую оценку истинной дисперсии генеральной совокупности.

**var**(**A**, **B**, **C**, ...) — возвращает значение дисперсии генеральной совокупности, вычисленное по формуле $\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}(\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M}))^2 \Big/ \mathbf{m}\cdot\mathbf{n}$. Таким образом, $\mathbf{var}\cdot\frac{\mathbf{m}\cdot\mathbf{n}}{\mathbf{m}\cdot\mathbf{n}-1}=\mathbf{Var}$.

***Функции статистических распределений и генераторов случайных чисел***

Обозначение следующих функций выполнены следующим образом:

**d...** — функция плотности распределения вероятности (вероятность того, что случайная величина примет значение **x**);

**p...** — функция кумулятивного распределения вероятностей (вероятность того, что случайная величина примет значение $\mathbf{X} \le \mathbf{x}$);

**q...** — функция обратного кумулятивного распределения вероятностей — квантиль (такое значение **x**, при котором вероятность не превышает заданное значение **p**);

**r...** — функция, возвращающая вектор из **m** случайных чисел, которые имеют указанное распределение "**...**".

***β-распределение***,

$$\frac{\Gamma(\mathbf{s1}+\mathbf{s2})}{\Gamma(\mathbf{s1})\cdot\Gamma(\mathbf{s2})}\cdot\mathbf{x}^{\mathbf{s1}-1}\cdot(1-\mathbf{x})^{\mathbf{s2}-1},$$

при параметрах формы $(\mathbf{s1},\mathbf{s2})>0$ и $0 < \mathbf{x} < 1$: **dbeta**(**x**, **s1**, **s2**); **pbeta**(**x**, **s1**, **s2**); **qbeta**(**p**, **s1**, **s2**); **rbeta**(**m**, **s1**, **s2**).

***Биномиальное распределение***,

$$\frac{\mathbf{n!}}{\mathbf{k!}\cdot(\mathbf{n}-\mathbf{k})\mathbf{!}}\cdot\mathbf{p}^{\mathbf{k}}\cdot(1-\mathbf{p})^{\mathbf{n}-\mathbf{k}},$$

где **n** — количество независимых испытаний с вероятностью успеха **p**, $0 \le \mathbf{p} \le 1$; **q** — вероятность успеха при однократном испытании; **k** — количество успехов, $0 \le \mathbf{k} \le \mathbf{n}$: **dbinom**(**k**, **n**, **p**); **pbinom**(**k**, **n**, **p**); **qbinom**(**p**, **n**, **q**); **rbinom**(**m**, **n**, **p**).

***Распределение Коши***,

$$\left[\pi\cdot\mathbf{s}\cdot\left[1+\left(\frac{\mathbf{x}-\mathbf{a}}{\mathbf{s}}\right)^2\right]\right]^{-1},$$

где **a** — параметр размещения; **s** > 0 — параметр масштаба: **dcauchy**(**x**, **a**, **s**); **pcauchy**(**x**, **a**, **s**); **qcauchy**(**p**, **a**, **s**); **rcauchy**(**m**, **a**, **s**).

***Распределение*** $\chi^2$,

$$\frac{\exp(-\mathbf{x}/2)}{2\cdot\Gamma(\mathbf{d}/2)}\cdot\left(\frac{\mathbf{x}}{2}\right)^{\mathbf{d}/2-1},$$

где **d** — параметр формы (количество степеней свободы), **d** > 0; **x** > 0: **dchisq**(**x**, **d**); **pchisq**(**x**, **d**); **qchisq**(**p**, **d**); **rchisq**(**m**, **d**).

***Экспонентное (показательное) распределение***,

$$\mathbf{r}\cdot\exp(-\mathbf{r}\cdot\mathbf{x}),$$

где **r** — параметр масштаба, **r** > 0; **x** > 0: **dexp**(**x**, **r**); **pexp**(**x**, **r**); **qexp**(**p**, **r**); **rexp**(**m**, **r**).

***F-распределение Фишера***,

$$\frac{\sqrt{\mathbf{d1}^{\mathbf{d1}}\cdot\mathbf{d2}^{\mathbf{d2}}}\cdot\Gamma\left(\frac{\mathbf{d1}+\mathbf{d2}}{2}\right)}{\Gamma\left(\frac{\mathbf{d1}}{2}\right)\cdot\Gamma\left(\frac{\mathbf{d2}}{2}\right)}\cdot\sqrt{\frac{\mathbf{x}^{\mathbf{d1}-2}}{(\mathbf{d1}+\mathbf{d2}\cdot\mathbf{x})^{\mathbf{d1}+\mathbf{d2}}}},$$

где **d1**,**d2** > 0 — количество степеней свободы; **x** > 0: **dF**(**x**, **d1**, **d2**); **pF**(**x**, **d1**, **d2**); **qF**(**p**, **d1**, **d2**); **rF**(**m**, **d1**, **d2**).

***γ-распределение***,

$$\mathbf{x}^{\mathbf{s}-1}\cdot\exp(-\mathbf{x})/\Gamma(\mathbf{s}),$$

где **s** > 0 — параметр формы; **x** ≥ 0: **dgamma**(**x**, **s**); **pgamma**(**x**, **s**); **qgamma**(**p**, **s**); **rgamma**(**m**, **s**).

***Геометрическое распределение***,

$$\mathbf{p}\cdot(1-\mathbf{p})^{\mathbf{n}},$$

где 0 < **p** < 1 — вероятность успеха; **n** — количество испытаний, **n** ≥ 0: **dgeom**(**n**, **p**); **pgeom**(**n**, **p**); **qgeom**(**p**, **q**); **rgeom**(**m**, **p**).

***Гипергеометрическое распределение***,

$$\mathbf{C}_{\mathbf{a}}^{\mathbf{m}}\cdot\frac{\mathbf{C}_{\mathbf{b}}^{\mathbf{n}-\mathbf{m}}}{\mathbf{C}_{\mathbf{a}+\mathbf{b}}^{\mathbf{n}}}=\mathbf{combin}(\mathbf{a},\mathbf{m})\cdot\frac{\mathbf{combin}(\mathbf{b},\mathbf{n}-\mathbf{m})}{\mathbf{combin}(\mathbf{a}+\mathbf{b},\mathbf{n})},$$

где **a** + **b** — популяция элементов, из которых **a** имеют некоторое свойство (извлечение такого элемента считается успехом); **n** — объем выборки из популяции без возвращения; **m** — количество успехов в выборке. Необходимое условие $\mathbf{max}\{0,\mathbf{n}-\mathbf{b}\}\le\mathbf{m}\le\mathbf{min}\{\mathbf{n},\mathbf{a}\}$: **dhypergeom**(**m**, **a**, **b**, **n**) и **phypergeom**(**m**, **a**, **b**, **n**), для которых 0 ≤ **m** ≤ **a**, 0 ≤ (**n**–**m**) ≤ **b**, 0 ≤ **n** ≤ (**a**+**b**); **qhypergeom**(**p**, **a**, **b**, **n**) и **rhypergeom**(**m**, **a**, **b**, **n**), для которых 0 ≤ **p** < 1, **m** > 0, (**a**, **b**) ≥ 0, 0 ≤ **n** ≤ (**a**+**b**).

***Логнормальное распределение***,

$$\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma\cdot x}\cdot\exp\left[-\frac{1}{2\cdot\sigma^2}\cdot(\ln(x)-\mu)^2\right],$$

где $\mu$ — натуральный логарифм от среднего значения; $\sigma > 0$ — параметр формы (натуральный логарифм среднеквадратического отклонения); $x > 0$: **dlnorm**(**x**, $\mu$, $\sigma$); **plnorm**(**x**, $\mu$, $\sigma$); **qlnorm**(**p**, $\mu$, $\sigma$); **rlnorm**(**m**, $\mu$, $\sigma$).

***Логистическое распределение***,

$$\exp\left(\frac{a-x}{s}\right)\Big/\left[s\cdot\left(1+\exp\left(\frac{a-x}{s}\right)\right)^2\right],$$

где **a** — параметр размещения; **s** > 0 — параметр масштаба: **dlogis**(**x**, **a**, **s**); **plogis**(**x**, **a**, **s**); **qlogis**(**p**, **a**, **s**); **rlogis**(**m**, **a**, **s**).

***Отрицательное биномиальное распределение***,

$$C_{n+k-1}^{k}\cdot p^n\cdot(1-p)^k = \mathbf{combin}(n+k-1,k)\cdot p^n\cdot(1-p)^k,$$

где $0 < p \le 1$ – вероятность успеха; $x > 0$ – количество успехов; $k \ge 0$ – количество неудач: **dnbinom**(**k**, **n**, **p**); **pnbinom**(**k**, **n**, **p**); **qnbinom**(**p**, **n**, **q**); **rnbinom**(**m**, **n**, **p**).

***Нормальное (гауссово) распределение***,

$$\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\cdot\exp\left[-\frac{1}{2\cdot\sigma^2}\cdot(x-\mu)^2\right],$$

где $\mu$ — параметр размещения (математическое ожидание); $\sigma > 0$ — параметр масштаба (среднеквадратическое отклонение): **dnorm**(**x**, $\mu$, $\sigma$); **pnorm**(**x**, $\mu$, $\sigma$); **qnorm**(**p**, $\mu$, $\sigma$); **rnorm**(**m**, $\mu$, $\sigma$); **cnorm**(**x**) — то же, что и функция **pnorm** при $\mu = 0$ и $\sigma = 1$ (стандартизированное нормальное распределение).

***Распределение Пуассона***,

$$\lambda^k\cdot\exp(-\lambda)/k!,$$

где $\lambda > 0$ — параметр распределения; $k \ge 0$: **dpois**(**k**, $\lambda$); **ppois**(**k**, $\lambda$); **qpois**(**k**, $\lambda$); **rpois**(**m**, $\lambda$).

***t-распределение Стьюдента***,

$$\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)\cdot\sqrt{\pi\cdot d}}\cdot\left[\sqrt{\left(1+\frac{x^2}{d}\right)^{d+1}}\right]^{-1},$$

где **d** > 0 — количество степеней свободы; **x** — действительное число: **dt**(**x**, **d**); **pt**(**x**, **d**); **qt**(**p**, **d**); **rt**(**m**, **d**).

***Равномерное (прямоугольное) распределение***,

$$(b-a)^{-1},$$

где **a** и **b** — соответственно, нижняя и верхняя границы области значений, причем $a < b$, $a \le x \le b$: **dunif**(**x**, **a**, **b**); **punif**(**x**, **a**, **b**); **qunif**(**x**, **a**, **b**); **runif**(**m**, **a**, **b**); **rnd**(**x**) — возвращает равномерно распределенное случайное число на интервале между 0 и **x**.

***Распределение Вейбулла***,

$$\mathbf{s}\cdot\mathbf{x}^{\mathbf{s}-1}\cdot\exp\left(-\mathbf{x}^{\mathbf{s}}\right),$$

где $\mathbf{s} > 0$ — параметр формы; $\mathbf{x} > 0$: **dweibull**(**x**, **s**); **pweibull**(**x**, **s**); **qweibull**(**p**, **s**); **rweibull**(**m**, **s**).

**Seed**(**x**) — сбрасывает значение начального числа при генерации ПСЧ в **x** и возвращает предшествующее значение. Причем $1 \le \mathbf{x} \le 2147483647$, целое.

***Функции комбинаторного анализа и теории чисел***

**combin**(**n**, **k**) — возвращает количество подмножеств (комбинаций), каждая размерностью **k**, которая может быть образована из **n** объектов — эквивалент $\frac{\mathbf{n!}}{\mathbf{k!}\cdot(\mathbf{n}-\mathbf{k})!}$, где $\mathbf{n} \ge \mathbf{k}$; $\mathbf{n}, \mathbf{k} \ge 0$.

**gcd**(**A**, **B**, **C**, ...) — возвращает наибольший общий делитель — наибольшее число, на которое без остатка делятся каждый из аргументов **A**, **B**, **C**, ... .

**lcm**(**A**, **B**, **C**, ...) — возвращает наименьшее общее кратное — наименьшее положительное целое, которое без остатка делится на каждый из аргументов **A**, **B**, **C**, ... .

**mod**(**x**, **y**) — возвращает остаток от деления **x** на **y** (аргументы должны быть действительными, $\mathbf{y} \ne 0$; результат имеет тот же знак, что и **x**)

**permut**(**n**, **k**) — возвращает количество возможных способов выбора (перестановки) **k** разных объектов из **n** существующих — эквивалент $\mathbf{n!}/(\mathbf{n}-\mathbf{k})\mathbf{!}$, где $\mathbf{n} \ge \mathbf{k}$; $\mathbf{n}, \mathbf{k} \ge 0$.

***Функции интерполяции и экстраполяции***

**cspline**(**vx**, **vy**) — возвращает вектор **vs** вторых производных для данных, представленных векторами **vx** и **vy**. Построенная по этому вектору сплайновая кривая является кубической в конечных точках.

**interp**(**vs**,**vx**,**vy**,**x**) — возвращает значение сплайн-интерполированной функции для векторов **vx** (упорядоченного по возрастанию) и **vy** опорных точек и аргумента **x** на основе вектора **vs**. Векторы **vx** и **vy** имеют одинаковую размерность.

**linterp**(**vx**, **vy**, **x**) — возвращает значение линейно-интерполированной функции для заданных векторов **vx** (упорядоченного по возрастанию) и **vy** опорных точек и заданного аргумента **x**. Векторы **vx** и **vy** имеют одинаковую размерность. При этом точки на графике, полученные по данным векторов, соединяются отрезками прямых.

**lspline**(**vx**, **vy**) — то же, что и **cspline**, но сплайновая кривая в конечных точках является линейной.

**predict**(**v**, **m**, **n**) — возвращает вектор из **n** линейно прогнозированных значений на основе **m** последовательных элементов из вектора данных **v**

**pspline**(**vx**, **vy**) — то же, что и **cspline**, но сплайновая кривая в конечных точках является параболической.

***Функции регрессии***

**expfit**(**vx**, **vy**, [**vg**]) — экспонентная регрессия данных, определенных векторами **vx** и **vy**. Вектор **vg**, если используется, содержит прогнозированные значения параметров **a**, **b** и **c** в степенном уравнении $\mathbf{a}\cdot\mathbf{exp}(\mathbf{b}\cdot\mathbf{x})+\mathbf{c}$.

**genfit**(**vx**,**vy**,**vg**,**F**) — возвращает вектор **K** параметров функций **F**, которые дают минимальную среднеквадратическую погрешность описания функцией $\mathbf{F}(\mathbf{x}, \mathbf{K}_1, ..., \mathbf{K}_n)$ первичных данных **vx**, **vy**. Вектор **vg** должен содержать прогнозированные значения **n**-параметров.

**intercept**(**vx**, **vy**) — возвращает значение свободного члена **a** линейной регрессии $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$.

**lgsfit**(**vx**, **vy**, **vg**) — регрессия логистической функцией $\mathbf{a}/(1+\mathbf{b}\cdot\mathbf{exp}(-\mathbf{c}\cdot\mathbf{x}))$.

**line**(**vx**, **vy**) — возвращает вектор коэффициентов линейной регрессии функцией $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$.

**linfit**(**vx**, **vy**, **F**) — возвращает вектор коэффициентов линейной регрессии общего вида. Вектор **F** должен содержать функции $\mathbf{F}_1(\mathbf{x})$, ..., $\mathbf{F}_n(\mathbf{x})$.

**lnfit**(**vx**, **vy**) — регрессия логарифмической функцией $\mathbf{a}\cdot\mathbf{ln}(\mathbf{x})+\mathbf{b}$.

**loess**(**Mx**, **My**, **sp**) — возвращает вектор коэффициентов **vs** для регрессии отрезками полиномов (используется в паре с функцией **interp**). Параметр **sp** указывает размер локальной области приближаемых данных.

**logfit**(**vx**, **vy**, **vg**) — регрессия логарифмической функцией $\mathbf{a}\cdot\mathbf{ln}(\mathbf{x}+\mathbf{b})+\mathbf{c}$.

**medfit**(**vx**, **vy**) — возвращает вектор коэффициентов линейной регрессии функцией $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$, используя медианную регрессию.

**pwrfit**(**vx**, **vy**, **vg**) — регрессия степенной функцией $\mathbf{a}\cdot\mathbf{x}^{\mathbf{b}}+\mathbf{c}$.

**regress**(**Mx**, **vy**, **n**) — возвращает вектор коэффициентов **vs** для полиномиальной регрессии при степени полинома **n** (используется в паре с функцией **interp**).

**sinfit**(**vx**, **vy**, **vg**) — регрессия синусоидой $\mathbf{a}\cdot\mathbf{sin}(\mathbf{x}+\mathbf{b})+\mathbf{c}$.

**slope**(**vx**, **vy**) — возвращает значение углового коэффициента **b** линейной регрессии $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$.

***Функции статистического сглаживания данных***

**ksmooth**(**vx**,**vy**,**b**) — возвращает **m**-мерный вектор сглаженных элементов **vy**, вычисленных на основе распределения Гаусса. **vx**, **vy** — **m**-мерные векторы действительных чисел. Параметр **b** — ширина окна сглаживания.

**medsmooth**(**vy**,**n**) — возвращает **m**-мерный вектор сглаженных элементов **vy**, вычисленных с использованием метода «скользящих медиан». Параметр **n** — ширина окна сглаживания.

**supsmooth**(**vx**,**vy**) — возвращает **m**-мерный вектор сглаженных элементов **vy**, вычисленных на основе адаптивной процедуры линейного сглаживания методом наименьших квадратов.

**expsmooth**(**B**, $\alpha$) — сглаживание данных массива **B** путем применения экспоненциального сглаживания с весовым коэффициентом $\alpha$.

***Функции дискретного преобразования***

**cfft**(**A**) — прямое комплексное преобразование Фурье для массива комплексных чисел **A** (возвращает массив той же размерности, что и **A**). Если **A** является вектором, то $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{A_k} \cdot \mathbf{exp}[\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$, где **n** — количество элементов вектора **A**.

**CFFT**(**A**) — то же, но при нормировании $1/\mathbf{n}$ и отрицательном аргументе экспоненты.

**fft**(**v**) — прямое БПФ для данных, записанных в векторе **v** в виде действительных чисел с $2^{\mathbf{n}}$ элементами, где **n** — целое число (возвращает вектор размерностью $2^{\mathbf{n}-1}+1$): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{v_k} \cdot \mathbf{exp}[\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$.

**FFT**(**v**) — то же, но при нормировании $1/\mathbf{n}$ и отрицательном аргументе экспоненты.

**icfft**(**B**) — обратное комплексное преобразование Фурье, которое отвечает **cfft** (возвращает массив той же размерности аргументу **B**): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{B_k} \cdot \mathbf{exp}[-\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$.

**ICFFT**(**B**) — то же, но при нормировании $1/\mathbf{n}$ и положительном аргументе экспоненты.

**ifft**(**u**) — обратное БПФ, которое отвечает **fft**. Вектор **u** имеет размерность $2^{\mathbf{n}-1}+1$, где **n** — целое число (возвращается вектор размерностью $2^{\mathbf{n}}$): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{u_k} \cdot \mathbf{exp}[-\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$.

**IFFT**(**u**) — то же, но при нормировании $1/\mathbf{n}$ и положительном аргументе экспоненты.

**wave**(**v**) — прямое вейвлет-преобразование (вектор **v** должен содержать $2^{\mathbf{n}}$ действительных элементов, где **n** — целое число).

**iwave**(**u**) — вектор обратного вейвлет-преобразования (**u** — вектор частотных данных вейвлет-спектра, размерностью $2^{\mathbf{n}}$).

***Функции решения уравнений***

**Given** — директива, открывающая блок решения системы уравнений или неравенств.

**Find**(**x**, **y**, ...) — вектор скалярных значений **x**, **y**, ..., дающих решение системы уравнений в блоке, объявленном директивой **Given** (количество возвращаемых значений равняется количеству аргументов).

**lsolve**(**M**, **v**) — возвращает вектор неизвестных **x**, дающих решение системы линейных алгебраических уравнений вида **M·x = v**.

**Maximize**(**f**, **x**, **y**, ...) — возвращает вектор значений аргументов, при которых функция **f** достигает максимума (возможно задание дополнительных условий в блоке с **Given**).

**Minerr** (**x1**, **x2**, ...) — вектор значений для **x1**, **x2**, ..., которые представляют решение системы уравнений в блоке, объявленном директивой **Given** с минимальной среднеквадратичной погрешностью.

**Minimize**(**f**, **x**, **y**, ...) — возвращает вектор значений аргументов, при которых функция **f** достигает минимума (возможно задание дополнительных условий в блоке с **Given**).

**polyroots**(**v**) — возвращает вектор всех корней полинома, коэффициенты которого содержаться в **v**.

**root**(**f**(**var**), **var**, [**a**, **b**]) — возвращает значение переменной **var**, при которой выражение **f** = 0. [**a**, **b**] — возможный интервал поиска корня.

***Функции ошибок (интегралы вероятностей)***

**erf**(**x**) — функция ошибок (функция Лапласа), $\frac{2}{\sqrt{\pi}}\cdot\int_0^{\mathbf{x}}\mathbf{exp}\left(-\mathbf{t}^2\right)\mathrm{d}\mathbf{t}$.

**erfc**(**x**) — комплементарная функция ошибок, $\frac{2}{\sqrt{\pi}}\cdot\int_{\mathbf{x}}^{\infty}\mathbf{exp}\left(-\mathbf{t}^2\right)\mathrm{d}\mathbf{t}$; $\mathbf{erfc}(\mathbf{x})=1-\mathbf{erf}(\mathbf{x})$.

***Экспоненциальная (показательная) и логарифмическая функции***

**exp**(**z**) — основание натурального логарифма (число **e**) в степени **z**.
**log**(**z**) — десятичный логарифм от аргумента **z**.
**log**(**z**, **b**) — логарифм от аргумента **z** по основанию **b**.
**ln**(**z**) — натуральный логарифм от аргумента **z**.

***Функции возврата типа выражения***

**IsArray**(**x**) — возвращает 1, если **x** — массив, и 0 в других случаях.
**IsScalar**(**x**) — возвращает 1, если **x** — скаляр, и 0 в других случаях.
**IsString**(**x**) — возвращает 1, если **x** — текстовая строка, и 0 в других случаях.

***Функции импортирования и экспортирования файлов***

**APPENDPRN**(*file*) — дописывает содержание массива в конец разграниченного ASCII-файла. Количество столбцов в массиве должно соответствовать количеству столбцов в существующем файле.

**GETWAVINFO**(*file*) — возвращает вектор, который содержит следующие характеристики WAV-файла: количество каналов, частота дискретизации, количество бит на уровень квантования и среднее количество байт на секунду.

**READRGB**(*file*) — возвращает массив, состоящий из трех подмассивов, которые представляют соответственно красную, зеленую и синюю компоненты цветного изображения (BMP, GIF, JPG, TGA, PBM, PGM, PPM или TIF формата), находящегося в *file*.

**READ_RED**(*file*)
**READ_GREEN**(*file*)
**READ_BLUE**(*file*) — массивы, отвечающие соответственно красной, зеленой и синей компонентам объекта (изображения), которое содержится в *file*.

**READ_HLS**(*file*) — массив, который представляет данные о цвете объекта в *file* (оттенки цвета, яркость и интенсивность).

**READ_HLS_HUE**(*file*) — массивы, представляющие данные соответственно об оттенках,

**READ_HLS_LIGHT**(*file*) яркости и

**READ_HLS_SAT**(*file*) интенсивности цвета для объекта в *file*.

**READ_HSV**(*file*) — массив, который представляет значение оттенка цвета, яркости и насыщенности для компонента в *file*.

**READ_HSV_HUE**(*file*) — массивы, которые представляют значение соответствующего оттенка,

**READ_HSV_SAT**(*file*) насыщенности и

**READ_HSV_VALUE**(*file*) интенсивности цвета для компонента в *file*.

**READ_IMAGE**(*file*) — массив, представляющий BMP, GIF, JPG, TGA, PCX, PBM, PGM, PPM или TIF-файл градациями серого.

**READBIN**(*file*, *type*, [**t**]) — присвоение массиву двоичных значений из файла, тип (*type*) которого — *byte* (8 бит беззнаковое целое), *double* (64 бит с плавающей запятой) и др.

**READBMP**(*file*) — массив, который представляет объект BMP-формата в *file* градациями серого.

**READFILE**(*file*, *type*, [**t**]) — присвоение массиву данных из файла типа *type*. **t** — дополнительные параметры.

**READPRN**(*file*) — присвоение массиву значений из файла с именем *file.prn*.

**READWAV**(*file*) — массив, каждый столбец которого представляет собой отдельный канал, а каждая строка отвечает моменту времени, определяемому номером отсчета и частотой дискретизации сигнала.

**WRITERGB**(*file*) — запись цветного (16 млн. цветов) изображения в файл.

**WRITE_HSV**(*file*) — то же.

**WRITE_HLS**(*file*) — то же.

**WRITEBIN**(*file*, *type*, 1/0) — запись массива или скаляра в файл двоичных данных

**WRITEBMP**(*file*) — запись изображения в оттенках серого.

**WRITEPRN**(*file*) — запись данных (массива) в текстовый файл.

**WRITEWAV**(*file*, **f**, **Q**) — запись данных в звуковой файл с частотой дискретизации **f** и количеством бит на уровень квантования **Q**.

***Тригонометрические и гиперболические функции***

**a...**(**z**) — обратная тригонометрическая или гиперболическая функция "**...**" от аргумента **z**.

**...h**(**z**) — гиперболическая функция "**...**".

**cos**(**z**) — косинус.

**csc**(**z**) — косеканс.

**cot**(**z**) — котангенс.

**sec**(**z**) — секанс.

**sin**(**z**) — синус.

**tan**(**z**) — тангенс.

***Функции округления числа***

**ceil**(**z**) — возвращает наименьшее целое, превышающее или равняющееся **z**.

**Ceil**(**z**, **y**) — возвращает наименьшее число кратное **y**, такое что ≥ **z** (при **y** ≠ 0).

**floor**(**z**) — возвращает наибольшее целое число, которое является меньшим или равняется **z**.

**Floor**(**z**, **y**) — возвращает наибольшее число кратное **y**, такое что ≤ **x** (при **y** ≠ 0).

**round**(**z**, [**n**]) — возвращает аргумент **z**, округленный до **n** (опционально) разрядов десятичной дроби. Если **n** опущено, возвращает аргумент **z**, округленный к ближайшему целому (то есть считается, что **n** = 0). При этом функция аналогична **floor**(**z**), если дробная часть меньше 0.5, и **ceil**(**z**) в противном случае. Если **n** < 0, возвращает аргумент **z**, округленный до **n** знаков слева от десятичной запятой.

**Round**(**z**, **y**) — округляет **z** к ближайшему кратному **y** путем вычисления выражения **round**(**z**/**y**)·**y**.

**trunc**(**z**) — возвращает целую часть числа **z** путем отбрасывания мантиссы.

**Trunc**(**z**, **y**) — возвращает результат вычисления выражения **trunc**(**z**/**y**)·**y**.

***Строковые функции***

**concat**(**S1**,**S2**, ...) — объединение строковых переменных **S1**, **S2**, ... путем добавления строки **S2** к концу строки **S1** и т.д.

**error**(**S**) — вывод сообщения **S** об ошибке.

**num2str**(**z**) — возвращает строку, чьи символы отвечают десятичному значению числа **z**.

**search**(**S**, **sub**, **m**) — совершает поиск подстроки **sub** в строке **S**, начиная с позиции **m**.

**str2num**(**Sn**) — преобразование строкового представления числа **Sn** в действительное число. Число может быть комплексным, иметь инженерную запись, формат двоичного числа и т.п.

**str2vec**(**S**) — преобразование строки символов в вектор их ASCII-кодов.

**strlen**(**S**) — возвращает количество символов в строке **S**.

**substr**(**S**, **n**, **m**) — начиная с позиции **n**, выделяет из строки **S** подстроку длиной **m** символов.

**vec2str**(**v**) — конвертирует элементы вектора **v** ASCII-кодов в символьную строку. Допустимые значения элементов вектора для MathCAD v.11 — все кроме 0, для MathCAD v.12 — 9, 10, 13, 32-255.

***Векторные и матричные функции***

*Функции объединения массивов*

**augment**(**A**, **B**, ...) — объединяет бок о бок массивы **A**, **B**, ... с одинаковым количеством строк слева направо.

**stack**(**A**, **B**, ...) — объединяет массивы **A**, **B**, ... с одинаковым количеством столбцов сверху вниз.

*Функция разделения массивов*

**submatrix**(**A**,**m**,**n**,**i**,**j**) — возвращает подмассив из массива **A**, который состоит из элементов, общих для строк от **m** до **n** и столбцов от **i** до **j**. Необходимое условие: **m** ≤ **n** и **i** ≤ **j**.

*Функции создания массивов*

**diag**(**v**) — возвращает диагональную матрицу, элементы главной диагонали которой представляют собой вектор **v**.

**identity**(**n**) — возвращает единичную матрицу размерностью **n**×**n**.

**matrix**(**m**, **n**, **f**) — возвращает матрицу, (**i**, **j**)-й элемент которой содержит значение предварительно заданной некоторой функции **f**(**i**, **j**), где **i** = 0, 1, ..., **m** и **j** = 0, 1, ..., **n**.

*Функции определения размерности массивов*

**cols**(**A**) — возвращает количество столбцов в массиве **A**.

**last**(**v**) — возвращает индекс последнего элемента вектора **v**.

**length**(**v**) — возвращает общее количество элементов в векторе **v**.

**rows**(**A**) — возвращает количество строк в массиве **A**.

*Функции определения экстремумов значений элементов массивов*

**max**(**A**, **B**, ...) — возвращает наибольший по значению элемент среди массивов **A**, **B**, ... . Если значения элементов комплексные, максимум возвращается отдельно для действительной и мнимой частей.

**min**(**A**, **B**, ...) — возвращает наименьший по значению элемент среди массивов **A**, **B**, ... . Если значения элементов комплексные, минимум возвращается отдельно для действительной и мнимой частей.

*Функции сортировки массивов*

**csort**(**A**, **n**) — сортировка массива **A** путем перестановки строк в порядке возрастания значений элементов в столбце **n**. Если массив содержит комплексные элементы, мнимая часть игнорируется (это касается и следующих функций сортировки).

**reverse**(**v**) — изменяет порядок следования элементов вектора **v** на противоположный.

**rsort**(**A**, **n**) — сортировка массива **A** путем перестановки столбцов в порядке возрастания значений элементов в строке **n**.

**sort**(**v**) — сортировка элементов вектора **v** в порядке возрастания их значений.

*Функции поиска*

**hlookup**(**z**,**A**,**r**) — проводит поиск в верхней строке массива **A** заданного значения **z**. В столбце, в котором найдено **z**, выбирается элемент в строке **r**. Если найдено несколько значений — возвращается вектор (справедливо для всех функций поиска).

**lookup**(**z**, **A**, **B**) — выполняет поиск в массиве **A** заданного значения **z**. В случае отыскания — возвращает значение соответствующего (по строке и столбцу) элемента массиву **B**.

**match**(**z**, **A**) — проводит поиск в массиве **A** элементов, которые имеют значение **z** и возвращает индекс (индексы) их позиций в **A**.

**vlookup**(**z**,**A**,**c**) — проводит поиск в левом столбце массива **A** заданного значения **z**. В строке, в которой найдено **z**, выбирается элемент в столбце **c**.

*Функции определения числа обусловленности матриц*

**cond1**(**M**) — возвращает число обусловленности квадратной матрицы **M**, вычисленное в норме $\mathbf{L_1}$ как $\mathbf{L_1}(\mathbf{M})\cdot\mathbf{L_1}(\mathbf{M}^{-1}) \xrightarrow{\text{MathCAD}} \mathbf{norm1}(\mathbf{M})\cdot\mathbf{norm1}(\mathbf{M}^{-1})$.

**cond2**(**M**) — то же, в норме $\mathbf{L_2}$.

**conde**(**M**) — то же, в евклидовой норме.

**condi**(**M**) — то же, в бесконечной норме.

*Функции определения нормы матриц*

**norm1**(**M**) — возвращает $\mathbf{L_1}$-норму как максимальное значение среди результатов суммирования абсолютных значений элементов столбцов матрицы:

$$\mathbf{L_1}(\mathbf{M}) = \max_{j}\sum_{i=1}^{m}\left|\mathbf{M}_{i,j}\right|.$$

**norm2**(**M**) — возвращает $\mathbf{L_2}$-норму как корень квадратный из наибольшего собственного значения матрицы — $\lambda$:

$$\mathbf{L_2}(\mathbf{M}) = \sqrt{\max(\lambda)} \xrightarrow{\text{MathCAD}} \sqrt{\max\left(\text{eigenvals}\left(\mathbf{M}\cdot\overline{\mathbf{M}^{\mathbf{T}}}\right)\right)},$$

где $\overline{\mathbf{M}^{\mathbf{T}}}$ — сопряженная транспонированная (возможен обратный порядок) матрица **M**.

**norme**(**M**) — возвращает евклидову $\mathbf{L_e}$-норму как корень квадратный из суммы квадратов всех элементов: $\mathbf{L_e}(\mathbf{M}) = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{M}_{i,j})^2}$.

**normi**(**M**) — возвращает бесконечную $\mathbf{L_\infty}$-норму как максимальное значение среди результатов суммирования абсолютных значений элементов строк матрицы: $\mathbf{L_\infty}(\mathbf{M}) = \max_{i}\sum_{j=1}^{m}\left|\mathbf{M}_{i,j}\right|$.

*Функции определения ранга и линейных свойств матриц*

**geninv**(**A**) — создание левой, обратной к **A** (имеющей полный ранг) матрицы, которая удовлетворяет уравнению $\mathbf{L}\cdot\mathbf{A} = \mathbf{E}$, где **E** — единичная матрица $\mathbf{n}\times\mathbf{n}$; **L** — матрица $\mathbf{n}\times\mathbf{m}$; **A** — матрица $\mathbf{m}\times\mathbf{n}$, при $\mathbf{m} \geq \mathbf{n}$. То есть, $\mathbf{L} = (\mathbf{A}^{\mathbf{T}}\cdot\mathbf{A})^{-1}\cdot\mathbf{A}^{\mathbf{T}}$. Если **A** — квадратная, несингулярная матрица, то $\mathbf{L} = \mathbf{A}^{-1}$.

**rank**(**A**) — возвращает ранг (количество линейно независимых столбцов) массива **A**.

**rref**(**A**) — возвращает массив — ступенчатую форму массива **A** с сокращенным количеством строк.

**tr**(**M**) — возвращает след (сумму диагональных элементов) квадратной матрицы **M**.

*Функции определения собственных векторов и собственных значений матриц*

**eigenvals**(**M**) — возвращает вектор $\lambda$ собственных значений квадратной матрицы **M**, каждый элемент которого удовлетворяет равенству $\mathbf{M}\cdot\mathbf{V}^{<i>} = \lambda_i\cdot\mathbf{V}^{<i>}$, где $\mathbf{V}^{<i>}$ — **i**-й собственный вектор матрицы **M**. Суммирование всех элементов вектора $\lambda$ эквивалентно функции вычисления следа матрицы **M** — **tr**(**M**).

**eigenvec**(**M**, **z**) — возвращает нормированный собственный вектор $\mathbf{V}^{<i>}$ матрицы **M**, который отвечает ее собственному значению $\mathbf{z} = \lambda_i$.

**eigenvecs**(**M**) — возвращает матрицу **V**, столбцами которой являются собственные вектора матрицы **M** (порядок размещения собственных векторов отвечает порядку собственных значений $\lambda_i$ — функция **eigenvals**).

**genvals**(**M**, **N**) — возвращает вектор обобщенных собственных значений $\lambda'_i$ матрицы **M**, который отвечает матричному выражению $\mathbf{M}\cdot\mathbf{V}'^{\langle i\rangle} = \lambda'_i\cdot\mathbf{N}\cdot\mathbf{V}'^{\langle i\rangle}$, где **M** и **N** — квадратные матрицы действительных элементов.

**genvecs**(**M**, **N**) — возвращает матрицу **V'**, столбцы которой содержат нормированные обобщенные собственные вектора (порядок размещения векторов отвечает порядку обобщенных собственных значений $\lambda'_i$, которые возвращаются функцией **genvals**).

*Функции разложения матриц*

**cholesky**(**M**) — реализует разложение матрицы по Холецкому (метод квадратных корней), возвращая нижне-треугольную матрицу **L** (все элементы над главной диагональю являются нулевыми), что удовлетворяет равенству $\mathbf{L}\cdot\mathbf{L}^T = \mathbf{M}$, где **M** — действительная, положительно определенная квадратная матрица. При вычислении используется только верхне-треугольная часть **M**.

**lu**(**M**) — реализует разложение квадратной матрицы **M** путем возвращения массива, содержащего три объединенные бок о бок квадратные матрицы **P**, **L** и **U**, одинаковой размерности с **M**. Данные матрицы удовлетворяют уравнению $\mathbf{P}\cdot\mathbf{M} = \mathbf{L}\cdot\mathbf{Q}$, где **L** и **U** — соответственно нижне- и верхне-треугольные матрицы. Для выделения **P**, **L** и **U** из общего массива необходимо использовать функцию **submatrix** (это же касается функций **qr** и **svd**).

**qr**(**A**) — реализует разложение массива **A** размерностью $\mathbf{m}\times\mathbf{n}$ путем возвращения массива, чьи первые **m** столбцов являются квадратной ортонормированной матрицей **Q**, имеющей одинаковое с **A** количество строк, а **n** следующих столбцов содержат верхне-треугольную матрицу **R**, удовлетворяющую уравнению $\mathbf{A} = \mathbf{Q}\cdot\mathbf{R}$.

**svd**(**A**) — реализует разложение массива **A** размерностью $\mathbf{m}\times\mathbf{n}$ ($\mathbf{m} \geq \mathbf{n}$) по его сингулярным числам. Возвращает массив, состоящий из размещенных одна над одной матриц **U** и **V**, которые удовлетворяют уравнению $\mathbf{A} = \mathbf{U}\cdot\mathbf{D}\cdot\mathbf{V}^T$, где матрицы **U** — верхняя, размерностью $\mathbf{m}\times\mathbf{n}$; **V** — нижняя, размерностью $\mathbf{n}\times\mathbf{n}$; **D** — диагональная, на диагонали которой размещены сингулярные числа массива **A**, полученные с помощью функции **svds**: $\mathbf{D} \xrightarrow{\text{MathCAD}} \mathbf{diag}(\mathbf{svds}(\mathbf{A}))$.

**svds**(**A**) — возвращает вектор сингулярных чисел (корень квадратный от собственных значений матрицы $\mathbf{A}^T\cdot\mathbf{A}$) массива **A**. Последний может иметь размерность $\mathbf{m}\times\mathbf{n}$, $\mathbf{m} \geq \mathbf{n}$.

*Функции получения логарифмически разнесенных точек*

**logspace**(**min**, **max**, **npts**) — возвращает вектор из **npts** логарифмически разнесенных точек, начиная с **min** и заканчивая **max**, где **min** и **max** — положительные действительные числа.

**logpts**(**minexp**, **dec**, **dnpts**) — возвращает вектор, охватывающий **dec** декад из **dnpts** равноотдаленных точек, начиная с $10^{\mathbf{minexp}}$, по **dnpts** точек на декаду. Количество элементов результирующего вектора — $\mathbf{dec}\cdot\mathbf{dnpts}$.

*Функция вычисления корреляции векторов*

**correl**(**vx**, **vy**) — возвращает одномерный коэффициент корреляции векторов **vx** и **vy**. Результирующий вектор имеет длину **length**(**vx**)+**length**(**vy**)−1, где каждый элемент является результатом векторного произведения вектора **vx** со сдвинутой версией **vy**.

## ***Некоторые дополнительные функции обработки массивов изображения***

**addnoise**(**A**, **p**, **n**) — возвращает массив **A** с аддитивным шумом, причем к значению пикселя добавляется или отнимается значение **n** ($0 \le |\mathbf{n}| \le 255$) с вероятностью $\mathbf{p}/2$ ($0 \le \mathbf{p} \le 1$).

**and**(**A**, **B**) / **or**(**A**, **B**) — возвращает результат выполнения булевых операций "и"/"или" над массивами изображений **A** и **B** (последние должны иметь одинаковую размерность).

**augment3**(**R**, **G**, **B**) — назначение аналогично функции **augment**, но обязательно наличие трех аргументов

**binarize**(**A**, **thresh**) — возвращает двоичную версию массива **A**: пикселям, значения которых превышают порог **thresh**, присваивается значение 1, остальным — 0.

**binarize_auto**(**A**) — то же, но значение порога выбирается программой автоматически.

**binarize2**(**A**, **t1**, **t2**, **x**, **y**) — возвращает двоичную версию массива **A**: пикселям, значения которых не выходят за пределы [**t1**, **t2**], присваивается значение **x**, остальным — **y**.

**blend**(**A**, **B**) — возвращает смесь соразмерных массивов **A**, **B** путем попиксельного вычисления $\mathbf{A}_{i,j} + \mathbf{B}_{i,j} - (\mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j}/255)$.

**canny**(**A**, σ, **low**, **high**) — возвращает двоичное контурное изображение, полученное в результате выделения контуров изображения по алгоритму Кэнни. $\sigma > 0$ — среднеквадратическое отклонение (обычно, не более 2). **low** < **high** — гистерезисные пороги (действительные числа).

**center**(**A**) — возвращает результат ДПФ массива **A**, трансформированный таким образом, что постоянная составляющая находится в центре.

**centsmooth**(**A**) — возвращает массив **A**, сглаженный с помощью центрально взвешенного элемента размерностью 3×3.

**clip**(**A**, **Min**, **Max**) — возвращает массив **A**, значения элементов которого ограничены порогами **Min** и **Max**.

**colgrad**(**A**) — возвращает постолбцовый градиент (отличие между столбцами) массива **A**.

**dct2d**(**A**) — возвращает двумерное быстрое ДКП массива **A**.

**distform**(**A**, **fg**) — трансформация массива **A** по евклидовому расстоянию для оттенка серого **fg**.

**equalize**(**A**) — возвращает массив **A** изображения с интенсивностью пикселей шкалы уровней серого, урегулированной таким образом, чтобы формировать линейную совокупную гистограмму.

**extract**(**A**, **n**) — выделяет подмассив **n**-го цветового компонента (**n** = $1_R$, $2_G$, $3_B$) из трехкомпонентного массива **A**.

**funmap**(**A**, **f**) — применяет функцию одной переменной **f** к каждому элементу массива **A**.

**getnoise**(**A**) — возвращает отличие между первичным и медианно-отфильтрованным с помощью функции **medfilt** массивом **A**.

**hist2d**(**A**, **B**, **n**) — возвращает двумерную **n**-столбцовую цветовую гистограмму для массивов **A** и **B** одинаковой размерности.

**horzflip**(**A**) — возвращает зеркальное отображение вокруг горизонтальной оси массива **A**.

**idct2d**(**A**) — возвращает обратное двумерное быстрое ДКП массива **A**.

**imhist**(**A**, **n**) — возвращает **n**-столбцовую цветовую гистограмму массива **A** для значений между 0 и 255 включительно (значения, которые выходят за этот диапазон, игнорируются).

**imhist2**(**A**, **n**) — возвращает **n**-столбцовую цветовую гистограмму массива **A** для присутствующего диапазона значений интенсивностей.

**immse**(**A**, **B**) — возвращает среднеквадратическую ошибку между массивами изображения **A** и **B**.

**imquant**(**A**, **n**) — возвращает квантованную версию массива **A**, которая содержит только **n** уровней серого между 0 и 255, равноотстоящих друг от друга.

**imquant2**(**A**, **v**) — возвращает квантованную версию **A**, которая содержит отмеченные в векторе **v** уровни серого.

**imsnr**(**A**, **B**) — возвращает отношение «сигнал/шум» между массивами изображения **A** и **B**.

**invert**(**A**) — возвращает инвертированный массив изображения **A** путем поэлементного вычитания от 255 значений интенсивностей всех пикселей.

**invert2**(**A**) — возвращает массив **A**, новое значение элементов которого вычисляется путем вычитания из **max**(**A**) значения текущего элемента и добавления к полученному результату **mix**(**A**).

**levelmap**(**A**, **v**) — возвращает массив **A** с интенсивностью, измененной на значение, которое имеет элемент вектора **v** ($0 \le \mathbf{v}_i \le 255$), индекс **i** которого отвечает первичной интенсивности пикселя. Размерность **v** не должна быть меньшей существующего максимального значения интенсивности по всему множеству пикселей изображения.

**mask**(**A**, **B**) — возвращает массив **A**, каждый элемент которого заменяется на нуль, если соответствующий элемент массива-маски **B** равняется 0.

**medfilt**(**A**) — возвращает медианно-отфильтрованный массив **A**.

**orthosmooth[5]**(**A**) — возвращает массив **A**, сглаженный с помощью ортогонально взвешенного элемента размерностью 3×3 (или 5×5 для **orthosmooth5**).

**putregion(A, B, r, c)** — встраивает массив **B** в массив **A**, начиная со строки **r** и столбца **c**. Необходимые условия при этом:

$$\mathbf{r} + \mathbf{rows}(\mathbf{B}) - 1 < \mathbf{rows}(\mathbf{A}) \text{ и } \mathbf{c} + \mathbf{cols}(\mathbf{B}) - 1 < \mathbf{cols}(\mathbf{A}).$$

**quantfilt**(**A**, **W**, **quant**) — возвращает квантильно-отфильтрованный массив **A**, используя при этом смежную матрицу **W** (произвольной размерности) и квантиль $0 \le \mathbf{quant} \le 1$.

**relerror**(**A**, **B**) — возвращает значение соответствующей ошибки между массивами изображения **A** и **B**.

**replace**(**A**, **B**, **n**) — возвращает трехкомпонентный массив **A** с измененным на матрицу **B** подмассивом **n**-го цветового компонента. Матрица **B** должна иметь одинаковое с **A** количество строк и треть от количества столбцов **A**.

**rotate**(**A**, $\alpha$) — поворачивает массив **A** на $\alpha$ градусов против хода часовой стрелки.

**rotate90[180,270]**(**A**) — поворачивает массив **A** на 90, 180 или 270° против хода часовой стрелки.

**rowgrad**(**A**) — возвращает строковый градиент (отличие между строками) массива **A**.

**scale**(**A**, **Min**, **Max**) — возвращает массив **A**, элементы которого нормированы в диапазоне [**Min**, **Max**].

**shape_features**(**A**) — возвращает матрицу моментов и особенностей отображения каждого отдельного пикселя изображения **A**.

**threshold**(**A**, **thresh**) — возвращает массив **A**, с присвоением элементам, значения которых являются меньшими порога **thresh**, нулевых значений. Если **thresh** — отрицательное число, нуль присваивается тем элементам, значения которых превышают значение модуля **thresh**.

**translate**(**A**, **r**, **c**, **pad**) — возвращает массив одной размерности с **A**, причем оригинальные элементы **A** сдвинуты на **r** строк и **c** столбцов. Незаполненные элементы нового массива, появившиеся при этом, приобретают интенсивности $0 \le \mathbf{pad} \le 255$. Знак **r** и **c** определяет направление сдвига (плюс — вниз или вправо, минус — вверх или влево).

**unismooth[5]**(**A**) — возвращает массив **A**, сглаженный с помощью равномерно взвешенного элемента размерностью 3×3 (или 5×5 для **unismooth5**).

**vertflip**(**A**) — зеркальное отображение вокруг вертикальной оси массива **A**.

**warp**(**A**, **T**) — применяет билинейную деформацию изображения, представленного массивом **A**, используя точки привязки, которые содержатся в массиве **T**. Последний имеет размерность 8×2 и содержит, таким образом, 4 пары точек.

**zoom**(**A**, **hsc**, **vsc**) — возвращает массив **A**, промасштабированный с коэффициентами **hsc**, **vsc** (соответственно, по горизонтали и вертикали). Последние должны быть положительными, ненулевыми действительными числами.

***Некоторые дополнительные функции обработки массивов сигналов***

**ccepstrum**(**A**) — возвращает массив — результат вычисления комплексного кепстра многоканального сигнала **A**.

**cepstrum**(**v**) — возвращает кепстр вектора **v**.

**chirpz**(**A**, **lo**, **hi**, **d**) — возвращает частотный спектр сигнала **A** между **lo** и **hi** с интервалом частот **d**, причем $0 \le$ **lo** $<$ **hi** $\le 0.5$, а (**hi** – **lo**)/**d** $\ge 2$.

**costr**(**A**) — выполняет косинусное преобразование массива **A**, содержащего четное количество элементов.

**icostr**(**A**) — выполняет обратное косинусное преобразование массива **A**.

**sintr**(**A**) — выполняет синусное преобразование массива **A**, содержащего четное количество элементов.

**isintr**(**A**) — выполняет обратное синусное преобразование массива **A**.

**phase**(**A**) — возвращает массив фаз, вычисленных на основе комплексных элементов массива **A**.

**mag**(**A**) — возвращает массив амплитуд, вычисленных на основе комплексных элементов массива **A**.

**makeri**(**Mag**, **Phase**) — возвращает массив комплексных чисел, вычисленных на основе амплитудных и фазовых элементов, содержащихся, соответственно, в соразмерных массивах **Mag** и **Phase**.

**snr**(**vx**, **vy**, **n**, **r**, [**w**]) — вычисляет отношение «сигнал/шум» для векторов сигнала **vx** и **vy**. Векторы делятся на 1 $<$ **n** $<$ **length**(**vx**) перекрывающихся сегментов с коэффициентом перекрытия $0 \le$ **r** $> 1$. Каждый сегмент обрабатывается окном с полосой **w**.

**whiten**(**n**) — возвращает вектор размерностью **n**, элементы которого представляют собой равномерно распределенный белый шум.

**convol**(**A**, **B**) — выполняет вычисление свертки массивов **A** и **B**, которые должны метить как минимум 2 элемента.

**deconvol**(**B**, **A**) — осуществляет обращение свертки — нахождение оригинала **B** на основе **A**.

## *Приложение С*

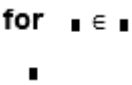
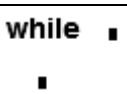
## Математические и системные константы MathCAD

Ниже приведены наименования математических и системных констант, правила их введения и назначения. Указанным константам можно присвоить и другие значения: или с помощью оператора присвоения непосредственно в рабочем документе, или же через диалоговое окно *Worksheet Options* из меню *Tools*. В скобках приведены общепринятые значения по умолчанию.

| *Константа* | *Введение* | *Назначение* |
|---|---|---|
| π | **[Ctrl][Shift]p** или **p[Ctrl]g** | Число π. В числовых расчетах MathCAD использует значение π с учетом 16 значащих цифр (3,1415926535897931). В символьных вычислениях π сохраняет свое точное значение |
| **e** | **e** | Основа натурального логарифма (2,718281828 4590451). В символьных вычислениях **e** сохраняет свое точное значение |
| ∞ | **[Ctrl][Shift]z** | Системная бесконечность ($10^{307}$) |
| % | **%** | Процент (0,01) |
| **1j** (**1i**) | **1j** (**1i**) | Комплексная единица ($\sqrt{-1}$) |
| **NaN** | | Не числовое значение |
| **ORIGIN** | | Нижняя граница индексации массивов (0) |
| **TOL** | | Погрешность числовых методов различных алгоритмов аппроксимации, которая определяет условия прекращения итераций числовым алгоритмом (0,001) |
| **CTOL** | | Погрешность сходимости в блоках решения уравнений, которая ограничивает невязку, задавая точность выполнения уравнений (0,001) |
| **Seed** | | Начальное число при генерации ПСЧ (1) |
| **PRNPRECISION** | | Количество десятичных знаков, используемых при записи файлов оператором **WRITEPRN** (4) |
| **PRNCOLWIDTH** | | Ширина столбца (количество символов), которая используется при записи файлов оператором **WRITEPRN** (8) |
| **CWD** | | Текстовая переменная, которая сохраняет адрес текущего (рабочего) документа на диске |
| **FRAME** | | Переменная счетчика кадров при работе с анимационными рисунками (0) |

## *Приложение D*

## Программные операторы MathCAD

| *Программный оператор* | *Шаблон* | *Назначение* |
|---|---|---|
| ***Add Line*** |  | Выполняет функции расширения программного модуля. Расширение фиксируется удлинением вертикальной черты программных модулей и их древовидным расширением. Благодаря этому можно создавать сколь угодно большие программы |
| ← |  | Оператор внутреннего (в теле программного модуля) локального присваивания |
| ***if*** |  | Оператор создания условных выражений:<br>*"выражение"* **if** *"условие"*<br>Если *"условие"* выполняется, то возвращается значение *"выражение"*. В паре с данным оператором обычно используются операторы **break** и **otherwise** |
| ***for*** |  | Оператор организации циклов с заданным количеством повторов:<br>**for** *"переменная"* ∈ **Nmin** .. **Nmax**<br>*"выражение"*<br>Если *"переменная"* изменяется, например, с шагом +1 от значения **Nmin** до **Nmax**, то *"выражение"* будет исполняться. Сама же *"переменная"* может быть использована в выражениях программного модуля |
| ***while*** |  | Оператор организации циклов, действующих до тех пор, пока выполняется некоторое *"условие"*:<br>**while** *"условие"*<br>*"выражение"* |
| ***otherwise*** |  | Оператор альтернативного условия. Обычно используется совместно с оператором **if** . Например, модуль<br>$f(x) := \begin{cases} 1 & \text{if } x < 0 \\ -1 & \text{otherwise} \end{cases}$<br>возвращает 1, если переменная **x** < 0, во всех остальных случаях возвращается значение –1 |
| ***break*** | **break** | Оператор вызова остановки работы программы (цикла). Обычно используется совместно с оператором условного выражения **if** и операторами циклов **for** и **while**, обеспечивая переход в конец тела цикла |

| *Программный оператор* | *Шаблон* | *Назначение* |
|---|---|---|
| ***continue*** | **continue** | Оператор продолжения. Применяется для продолжения работы после прерывания программы. Используется в паре с операторами задания циклов **for** и **while**, обеспечивая после прерывания переход в начало цикла |
| ***return*** | **return** ▪ | Оператор-функция возвращения. Прерывает выполнение программы и возвращает значение своего операнда-аргумента. Например, модуль<br>$f(x) := \left\| \begin{array}{l} \textbf{return } 20 \quad \textbf{if } x < 1 \\ 1 \end{array} \right.$<br>возвращает 20 при любом $x < 1$ , во всех остальных случаях возвращается значение 1 |
| ***on error*** | ▪ **on error** ▪ | Оператор обработки ошибок. Позволяет создавать конструкции обработчиков ошибок:<br>*"выражение 1"* **on error** *"выражение 2"*<br>Если при выполнении *"выражения 2"* возникает ошибка, выполняется *"выражение 1"* |

## *Приложение E*

## Таблица ASCII-кодов

| *DEC* | *BIN* | *Симв.* | *Описание* | *DEC* | *BIN* | *Симв.* | *Описание* |
|---|---|---|---|---|---|---|---|
| 0 | 00000000 | NuL | Не определен | 52 | 00110100 | 4 | Цифра четыре |
| 1 | 00000001 | ☺ | Начало заголовка | 53 | 00110101 | 5 | Цифра пять |
| 2 | 00000010 | ☻ | Начало текста | 54 | 00110110 | 6 | Цифра шесть |
| 3 | 00000011 | ♥ | Конец текста | 55 | 00110111 | 7 | Цифра семь |
| 4 | 00000100 | ♦ | Конец передачи | 56 | 00111000 | 8 | Цифра восемь |
| 5 | 00000101 | ♣ | Запрос | 57 | 00111001 | 9 | Цифра девять |
| 6 | 00000110 | ♠ | Подтвержд. приема | 58 | 00111010 | : | Двоеточие |
| 7 | 00000111 | • | Звуковой сигнал | 59 | 00111011 | ; | Точка с запятой |
| 8 | 00001000 | ◘ | Возвращение, забой | 60 | 00111100 | < | Знак меньше |
| 9 | 00001001 | ○ | Горизонт. табуляция | 61 | 00111101 | = | Знак равенства |
| 10 | 00001010 | ◙ | Перевод строки | 62 | 00111110 | > | Знак больше |
| 11 | 00001011 | ♂ | Вертик. табуляция | 63 | 00111111 | ? | Знак вопроса |
| 12 | 00001100 | ♀ | Перевод страницы | 64 | 01000000 | @ | Коммерческое "эт" |
| 13 | 00001101 | ♪ | Возврат каретки | 65 | 01000001 | A | Большая лат. *A* |
| 14 | 00001110 | ♫ | Верхний регистр | 66 | 01000010 | B | Большая лат. *B* |
| 15 | 00001111 | ☼ | Нижний регистр | 67 | 01000011 | C | Большая лат. *C* |
| 16 | 00010000 | ► | Отключение от линии | 68 | 01000100 | D | Большая лат. *D* |
| 17 | 00010001 | ◄ | Управление 1 | 69 | 01000101 | E | Большая лат. *E* |
| 18 | 00010010 | ↕ | Управление 2 | 70 | 01000110 | F | Большая лат. *F* |
| 19 | 00010011 | ‼ | Управление 3 | 71 | 01000111 | G | Большая лат. *G* |
| 20 | 00010100 | ¶ | Управление 4 | 72 | 01001000 | H | Большая лат. *H* |
| 21 | 00010101 | § | Нет подтверждения | 73 | 01001001 | I | Большая лат. *I* |
| 22 | 00010110 | ▬ | Синхронизация | 74 | 01001010 | J | Большая лат. *J* |
| 23 | 00010111 | ↨ | Конец перед. блока | 75 | 01001011 | K | Большая лат. *K* |
| 24 | 00011000 | ↑ | Отмена | 76 | 01001100 | L | Большая лат. *L* |
| 25 | 00011001 | ↓ | Конец носителя | 77 | 01001101 | M | Большая лат. *M* |
| 26 | 00011010 | → | Замена | 78 | 01001110 | N | Большая лат. *N* |
| 27 | 00011011 | ← | Прерывание | 79 | 01001111 | O | Большая лат. *O* |
| 28 | 00011100 | ∟ | Разделитель файлов | 80 | 01010000 | P | Большая лат. *P* |
| 29 | 00011101 | ↔ | Разделитель групп | 81 | 01010001 | Q | Большая лат. *Q* |
| 30 | 00011110 | ▲ | Разделитель записей | 82 | 01010010 | R | Большая лат. *R* |
| 31 | 00011111 | ▼ | Разделитель эл-тов | 83 | 01010011 | S | Большая лат. *S* |
| 32 | 00100000 |  | Пробел | 84 | 01010100 | T | Большая лат. *T* |
| 33 | 00100001 | ! | Знак восклицания | 85 | 01010101 | U | Большая лат. *U* |
| 34 | 00100010 | " | Двойные кавычки | 86 | 01010110 | V | Большая лат. *V* |
| 35 | 00100011 | # | Знак числа | 87 | 01010111 | W | Большая лат. *W* |
| 36 | 00100100 | $ | Знак доллара | 88 | 01011000 | X | Большая лат. *X* |
| 37 | 00100101 | % | Знак процента | 89 | 01011001 | Y | Большая лат. *Y* |
| 38 | 00100110 | & | Амперсанд | 90 | 01011010 | Z | Большая лат. *Z* |
| 39 | 00100111 | ' | Апостроф | 91 | 01011011 | [ | Левая квадр. скобка |
| 40 | 00101000 | ( | Левая круглая скобка | 92 | 01011100 | \ | Обр. косая черта |
| 41 | 00101001 | ) | Прав. круглая скобка | 93 | 01011101 | ] | Правая квадр. скобка |
| 42 | 00101010 | * | Звездочка | 94 | 01011110 | ^ | Знак вставки |
| 43 | 00101011 | + | Знак плюс | 95 | 01011111 | _ | Подчеркивание |
| 44 | 00101100 | , | Запятая | 96 | 01100000 | ` | Тупое ударение |
| 45 | 00101101 | - | Знак минус | 97 | 01100001 | a | Малая лат. *a* |
| 46 | 00101110 | . | Точка | 98 | 01100010 | b | Малая лат. *b* |
| 47 | 00101111 | / | Лев. косая черта | 99 | 01100011 | c | Малая лат. *c* |
| 48 | 00110000 | 0 | Цифра нуль | 100 | 01100100 | d | Малая лат. *d* |
| 49 | 00110001 | 1 | Цифра один | 101 | 01100101 | e | Малая лат. *e* |
| 50 | 00110010 | 2 | Цифра два | 102 | 01100110 | f | Малая лат. *f* |
| 51 | 00110011 | 3 | Цифра три | 103 | 01100111 | g | Малая лат. *g* |

| DEC | BIN | Симв. | Описание | DEC | BIN | Симв. | Описание |
|---|---|---|---|---|---|---|---|
| 104 | 01101000 | h | Малая лат. *h* | 160 | 10100000 | | Неразрывный пробел |
| 105 | 01101001 | i | Малая лат. *i* | 161 | 10100001 | Ў | Бол. кирил. *У кратн.* |
| 106 | 01101010 | j | Малая лат. *j* | 162 | 10100010 | ў | Мал. кирил. *у кратн.* |
| 107 | 01101011 | k | Малая лат. *k* | 163 | 10100011 | J | Большая кирил. *Йэ* |
| 108 | 01101100 | l | Малая лат. *l* | 164 | 10100100 | ¤ | Валюта |
| 109 | 01101101 | m | Малая лат. *m* | 165 | 10100101 | Ґ | Большая кирил. *Гэ* |
| 110 | 01101110 | n | Малая лат. *n* | 166 | 10100110 | ¦ | Разорванная черта |
| 111 | 01101111 | o | Малая лат. *o* | 167 | 10100111 | § | Знак параграфа |
| 112 | 01110000 | p | Малая лат. *p* | 168 | 10101000 | Ё | Большая кирил. *Ё* |
| 113 | 01110001 | q | Малая лат. *q* | 169 | 10101001 | © | Авторское право |
| 114 | 01110010 | r | Малая лат. *r* | 170 | 10101010 | Є | Большая кирил. *Є* |
| 115 | 01110011 | s | Малая лат. *s* | 171 | 10101011 | « | Левые угловые кавычки |
| 116 | 01110100 | t | Малая лат. *t* | 172 | 10101100 | ¬ | Угловое тире |
| 117 | 01110101 | u | Малая лат. *u* | 173 | 10101101 | - | Мягкий перенос |
| 118 | 01110110 | v | Малая лат. *v* | 174 | 10101110 | ® | Зарегистр. торг. знак |
| 119 | 01110111 | w | Малая лат. *w* | 175 | 10101111 | Ї | Большая кирил. *Ї* |
| 120 | 01111000 | x | Малая лат. *x* | 176 | 10110000 | ° | Градус |
| 121 | 01111001 | y | Малая лат. *y* | 177 | 10110001 | ± | "Плюс или минус" |
| 122 | 01111010 | z | Малая лат. *z* | 178 | 10110010 | I | Большая кирил. *І* |
| 123 | 01111011 | { | Левая фигурная скобка | 179 | 10110011 | і | Малая кирил. *і* |
| 124 | 01111100 | \| | Вертикальная черта | 180 | 10110100 | ґ | Малая кирил. *гэ* |
| 125 | 01111101 | } | Правая фигурн. скобка | 181 | 10110101 | µ | Знак микро (малая мю) |
| 126 | 01111110 | ~ | Тильда | 182 | 10110110 | ¶ | Абзац |
| 127 | 01111111 | ⌂ | Удаление | 183 | 10110111 | · | Серединная точка |
| 128 | 10000000 | Ђ | Большая кирил. *Дже* | 184 | 10111000 | ё | Малая кирил. *ё* |
| 129 | 10000001 | Ѓ | Большая кирил. *Ге* | 185 | 10111001 | № | Номер |
| 130 | 10000010 | ‚ | Нижний апостроф | 186 | 10111010 | є | Малая кирил. *є* |
| 131 | 10000011 | ѓ | Малая кирил. *ге* | 187 | 10111011 | » | Прав. угловые кавычки |
| 132 | 10000100 | „ | Двойные нижн. кавычки | 188 | 10111100 | j | Малая кирил. *йэ* |
| 133 | 10000101 | … | Горизонт. три точки | 189 | 10111101 | S | Большая кирил. *Зэ* |
| 134 | 10000110 | † | Начало выдел. данных | 190 | 10111110 | s | Малая кирил. *зэ* |
| 135 | 10000111 | ‡ | Конец выдел. данных | 191 | 10111111 | ї | Малая кирил. *ї* |
| 136 | 10001000 | € | Знак евро | 192 | 11000000 | А | Большая кирил. *А* |
| 137 | 10001001 | ‰ | Знак промилле | 193 | 11000001 | Б | Большая кирил. *Б* |
| 138 | 10001010 | Љ | Большая кирил. *Ле* | 194 | 11000010 | В | Большая кирил. *В* |
| 139 | 10001011 | ‹ | Левая угловая кавычка | 195 | 11000011 | Г | Большая кирил. *Г* |
| 140 | 10001100 | Њ | Большая кирил. *Не* | 196 | 11000100 | Д | Большая кирил. *Д* |
| 141 | 10001101 | Ќ | Большая кирил. *Ке* | 197 | 11000101 | Е | Большая кирил. *Е* |
| 142 | 10001110 | Ћ | Большая кирил. *Ше* | 198 | 11000110 | Ж | Большая кирил. *Ж* |
| 143 | 10001111 | Џ | Большая кирил. *Же* | 199 | 11000111 | З | Большая кирил. *З* |
| 144 | 10010000 | ђ | Малая кирил. *де* | 200 | 11001000 | И | Большая кирил. *И* |
| 145 | 10010001 | ‘ | Лев. одиночн. кавычка | 201 | 11001001 | Й | Большая кирил. *Й* |
| 146 | 10010010 | ’ | Прав. одиночн. кавычка | 202 | 11001010 | К | Большая кирил. *К* |
| 147 | 10010011 | “ | Левые кавычки | 203 | 11001011 | Л | Большая кирил. *Л* |
| 148 | 10010100 | ” | Правые кавычки | 204 | 11001100 | М | Большая кирил. *М* |
| 149 | 10010101 | • | Ожидание сообщения | 205 | 11001101 | Н | Большая кирил. *Н* |
| 150 | 10010110 | – | Короткое тире | 206 | 11001110 | О | Большая кирил. *О* |
| 151 | 10010111 | — | Длинное тире | 207 | 11001111 | П | Большая кирил. *П* |
| 152 | 10011000 | | Не определен | 208 | 11010000 | Р | Большая кирил. *Р* |
| 153 | 10011001 | ™ | Торговая марка | 209 | 11010001 | С | Большая кирил. *С* |
| 154 | 10011010 | љ | Малая кирил. *ле* | 210 | 11010010 | Т | Большая кирил. *Т* |
| 155 | 10011011 | › | Правя угловая кавычка | 211 | 11010011 | У | Большая кирил. *У* |
| 156 | 10011100 | њ | Малая кирил. *не* | 212 | 11010100 | Ф | Большая кирил. *Ф* |
| 157 | 10011101 | ќ | Малая кирил. *кє* | 213 | 11010101 | Х | Большая кирил. *Х* |
| 158 | 10011110 | ћ | Малая кирил. *ше* | 214 | 11010110 | Ц | Большая кирил. *Ц* |
| 159 | 10011111 | џ | Малая кирил. *же* | 215 | 11010111 | Ч | Большая кирил. *Ч* |

| DEC | BIN | Симв. | Описание | DEC | BIN | Симв. | Описание |
|---|---|---|---|---|---|---|---|
| 216 | 11011000 | Ш | Большая кирил. *Ш* | 236 | 11101100 | м | Малая кирил. *м* |
| 217 | 11011001 | Щ | Большая кирил. *Щ* | 237 | 11101101 | н | Малая кирил. *н* |
| 218 | 11011010 | Ъ | Большая кирил. *Ъ* | 238 | 11101110 | о | Малая кирил. *о* |
| 219 | 11011011 | Ы | Большая кирил. *Ы* | 239 | 11101111 | п | Малая кирил. *п* |
| 220 | 11011100 | Ь | Большая кирил. *Ь* | 240 | 11110000 | р | Малая кирил. *р* |
| 221 | 11011101 | Э | Большая кирил. *Э* | 241 | 11110001 | с | Малая кирил. *с* |
| 222 | 11011110 | Ю | Большая кирил. *Ю* | 242 | 11110010 | т | Малая кирил. *т* |
| 223 | 11011111 | Я | Большая кирил. *Я* | 243 | 11110011 | у | Малая кирил. *у* |
| 224 | 11100000 | а | Малая кирил. *а* | 244 | 11110100 | ф | Малая кирил. *ф* |
| 225 | 11100001 | б | Малая кирил. *б* | 245 | 11110101 | х | Малая кирил. *х* |
| 226 | 11100010 | в | Малая кирил. *в* | 246 | 11110110 | ц | Малая кирил. *ц* |
| 227 | 11100011 | г | Малая кирил. *г* | 247 | 11110111 | ч | Малая кирил. *ч* |
| 228 | 11100100 | д | Малая кирил. *д* | 248 | 11111000 | ш | Малая кирил. *ш* |
| 229 | 11100101 | е | Малая кирил. *е* | 249 | 11111001 | щ | Малая кирил. *щ* |
| 230 | 11100110 | ж | Малая кирил. *ж* | 250 | 11111010 | ъ | Малая кирил. *ъ* |
| 231 | 11100111 | з | Малая кирил. *з* | 251 | 11111011 | ы | Малая кирил. *ы* |
| 232 | 11101000 | и | Малая кирил. *и* | 252 | 11111100 | ь | Малая кирил. *ь* |
| 233 | 11101001 | й | Малая кирил. *й* | 253 | 11111101 | э | Малая кирил. *э* |
| 234 | 11101010 | к | Малая кирил. *к* | 254 | 11111110 | ю | Малая кирил. *ю* |
| 235 | 11101011 | л | Малая кирил. *л* | 255 | 11111111 | я | Малая кирил. *я* |

# СПИСОК ЛИТЕРАТУРЫ


1. Артёхин Б.В. *Стеганография* // Журнал “Защита информации. Конфидент”. — 1996. — №4. — С.47-50.
2. D. Kahn, *The Code-Breakers: The Story of Secret Writing*. MacMillan Publishing Company, New York, USA, 1996.
3. Хорошко В.О., Азаров О.Д., Шелест М.Є., Яремчук Ю.Є. *Основи комп’ютерної стеганографії* : Навчальний посібник для студентів і аспірантів. — Вінниця: ВДТУ, 2003. — 143 с.
4. Барсуков В.С., Романцов А.П. *Компьютерная стеганография: вчера, сегодня, завтра. Технологии информационной безопасности XXI века*. — материалы Internet-ресурса «Специальная техника» (http://st.ess.ru/).
5. Грибунин В.Г., Оков И.Н., Туринцев И.В. *Цифровая стеганография*. — М.: Солон-Пресс, 2002. — 272 с.
6. A. Kerckhoffs, *La Cryptographie Militaire*. Journal des sciences militaires, pp: 5–83, Jan. 1883, pp: 161-191, Feb. 1883.
7. Gustavus J. Simmons, *The Prisoner’s Problem And The Subliminal Channel*, Advances in Cryptology: Proceedings of Workshop on Communications Security (Crypto'83, David Chaum, ed.), Plenum Press. 1984. Pp.51-67.
8. Gustavus J. Simmons, *The History of Subliminal Channels*. // IEEE Journal on Selected Areas of Communications. 1998. Vol.16, №4. pp. 452-461.
9. J. Fridrich, R. Du, M. Long, *Steganalysis of LSB Encoding in Color Images*, Proceedings of ICME 2000, New York City, July 31 — August 2, New York, USA.
10. B. Pfitzmann, *Information Hiding Terminology*. In: Information Hiding, Springer Lecture Notes in Computer Science. V.1174. 1996. Pp.347-350.
11. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. New York // John Wiley and Sons, 1996.
12. S. Craver, On Public-Key Steganography in the Presence of an Active Warden // Technical report RC 20931, IBM, 1997. 13p.
13. Ross J. Anderson, *Stretching the Limits of Steganography*. In: Information Hiding, Springer Lecture Notes in Computer Science. V.1174. 1996. Pp.39-48.
14. W. Bender, D. Gruhl, N. Morimoto, A. Lu, *Techniques for Data Hiding*. IBM Systems Journal, 35(3&4): pp. 313-336, 1996.
15. C. Cachin, *An Information-Theoretic Model for Steganography*. In: Information Hiding — 2nd International Workshop, Springer as Lecture Notes in Computing Science, vol.1525, April 1998, pp.306-318.
16. P. Bassia, I. Pitas, *Robust Audio Watermarking In The Time Domain* // Department of Informatics, University of Thessalonica (http://poseidon.csd.auth.gr/voyatzis/creus.zip).
17. R. Popa, *An Analysis of Steganographic Techniques*. The Polytechnic University of Timisoara, Faculty of Automatics and Computers, Department of Computer Science and Software Engineering. 1998. 59p.

**18.** N.F. Johnson, S. Jajodia, *Steganalysis of Images Created using Current Steganography Software*, Workshop on Information Hiding Proceedings, Portland Oregon, Springer as Lecture Notes in Computer Science, vol.1525, 1998.

**19.** Husrev T. Sencar , Mahalingam Ramkumar, Ali N. Akansu, *Data Hiding Fundamentals And Applications. Content Security In Digital Multimedia*. ELSEVIER science and technology books, 2004. 364p.

**20.** Neil F. Johnson, Zoran Duric, Sushil Jajodia, *Information Hiding : Steganography and Watermarking. - Attacks and Countermeasures*. Kluwer Academic Publishers. 2001. 160p.

**21.** S. Katzenbeisser, Fabien A. P. Petitcolas (Editors), *Information Hiding Techniques for Steganography And Digital Watermark*. Artech House Publishers. 1999. 220p.

**22.** Генне О.В. *Основные положения стеганографии* // Журнал "Защита информации. Конфидент", №3, 2000.

**23.** Кустов В.Н., Федчук А.А. *Методы встраивания скрытых сообщений* // Журнал "Защита информации. Конфидент", №3, 2000, с.34.

**24.** Быков С.Ф. *Алгоритм сжатия JPEG с позиции компьютерной стеганографии* // "Защита информации. Конфидент", №3, 2000, с.26.

**25.** Очков В.Ф. *MathCAD 12 для студентов и инженеров*. — СПб.: BHV – Санкт-Петербург, 2005. — 464 с.

**26.** Дьяконов В.П. *Энциклопедия MathCAD 2001i и MathCAD 11*. — М.: Солон-Пресс, 2004. — 831 с.

**27.** Kefa Rabah, *Steganography — The Art of Hiding Data*. Information Technology Journal 3 (3): pp. 245-269. 2004.

**28.** D. Verton, *Experts Debate Biggest Network Security Threats*. USA Today, 12 Apr. 2002. (http://www.usatoday.com/life/cyber/tech/cw1.htm).

**29.** K. Maney, *Bin Laden's Messages Could Be Hiding In Plain Sight*. USA Today, 19 Dec. 2001. (http://www.usatoday.com/life/cyber/ccarch/2001/12/19/maney.htm).

**30.** R. Mohanakrishnan, *Steganography: Snake In the Grass*. The Hindu Newspaper (India), Dec. 6, 2001.

**31.** D.L. Schilling, ed., *Meteor Burst Communications: Theory And Practice*. Wiley series in telecommunications, New York, USA: Wiley, 1993.

**32.** John H. Nugent, Mahesh S. Raisinghani, The Information Technology And Telecommunications Security Imperative: Important Issues And Drivers. Journal of Electronic Commerce Research, Vol.3, №1, 2002.

**33.** Закон Украины "Про защиту информации в информационно-телекоммуникационных системах", №2594-IV от 31 мая 2005 года.

**34.** Анин Б.Ю. *Защита компьютерной информации*. — СПб.: BHV - Санкт-Петербург, 2000. — 284 с.

**35.** Соколов А.В., Шаньгин В.Ф. Защита информации в распределённых корпоративных сетях и системах. — М.: ДМК Пресс, 2002. — 656 с.

**36.** Хорошко В.О., Огаркова І.М., Чирков Д.В., Голего А.Г., Горохова Т.Б. *Термінологічний довідник з питань технічного захисту інформації*. / За ред. проф. Хорошка В.О. 3 вид., доп. і перероб. — К. : ТОВ "ПоліграфКонсалтинг", 2003. — 286 с.

**37.** Голубєв В.О., Гавловський В.Д., Цимбалюк В.С. Інформаційна безпека: проблеми боротьби зі злочинами у сфері використання комп'ютерних технологій. — Запоріжжя: Просвіта, 2001. — с.198-201.

**38.** Информационный ресурс исследовательской группы "CNews Analytics" (http://www.cnews.ru/).

**39.** Информационный ресурс исследовательского центра "DataPro Research Corporation". (http://www.datapro.com/).

**40.** N.F. Johnson, S. Jajodia, *Steganalysis: The Investigation of Hidden Information*, IEEE Information Technology Conference, Syracuse, New York, USA, Sept. 1st-3rd. 1998.

**41.** S. Voloshynovskiy, S. Pereira, V. Iquise, T. Pun, *Attack Modelling: Towards a Second Generation Watermarking Benchmark*. // Preprint. University of Geneva, 2001. 58p.

**42.** Грибунин В.Г. *Критерии оценки надёжности паролей*. — М.: РУСКАРД, 2003. (http://www.ruscard.org/).

**43.** Официальный сайт Национального института стандартов и технологий (НИСТ) США — http://www.nist.gov/.

**44.** M. Ramkumar, *Data Hiding in Multimedia*. PhD Thesis. New Jersey Institute of Technology, 1999. 72p.

**45.** C.E. Shannon, *A Mathematical Theory of Communication*. Bell System Technical Journal, 27 (1948), pp.379-423, 623-656.

**46.** Marvel L. *Image Steganography for Hidden Communication*. PhD Thesis. University of Delaware, 1999. 115p.

**47.** Cox J., Miller M., McKellips A. *Watermarking as Communications With Side Information* // Proceedings of the IEEE. 1999. Vol.87. №7. P.1127-1141.

**48.** Барсуков В.С. Стеганографические технологии защиты документов, авторских прав и информации // Обзор специальной техники. — 2000. — №2. — С.31-40.

**49.** M. Kutter and F.A.P. Petitcolas, *A Fair Benchmark For Image Watermarking Systems*. Electronic Imaging '99. Security and Watermarking of Multimedia Contents, vol. 3657, Sans Jose, CA, USA, 25-27 January 1999.

**50.** Khalid Sayood, *Introduction to Data Compression*, chapter 7, p. 142. Morgan Kaufmann Publishers, 1996.

**51.** Ahmet M. Eskicioglu and Paul S. Fisher. *Image Auality Measures And Their Performance*. IEEE Transactions on Communication, 43(12): 2959-2965, December 1995.

**52.** P.R.R.L. Nunes, A. Alcaim, and M.R.L. Fragoso da Silva. *Quality Measures of Compressed Images for Classification Purposes*. Technical Report CCR-146, IBM Brasil, Rio Scientific Center, P.O. Box 4624, 20.0001 Rio de Janeiro, Brazil, October 1992.

**53.** S. Winkler. *A Perceptual Distortion Metric for Digital Color Video*. In: SPIE Proceedings of Human Vision and Electronic Imaging, vol. 3644, San Jose, CA, January 1999.

**54.** S. Winkler. *A Perceptual Distortion Metric for Digital Color Images*. In: Proc. ICIP, vol. 3, pp. 399-403, Chicago, IL, October 1998.

**55.** C.J. van den Branden Lambrecht and J.E. Farrell. *Perceptual Quality Metric for Digitally Coded Color Images*. In: Proceeding of EUSIPCO, pp. 1175-1178, Trieste, Italy, September 1996.

**56.** S.J.P. Westen, R.L. Lagendijk, and J. Biemond. *Perceptual Image Quality Based on a Multiple Channel HVS Model*. In: Proceeding of ICASP, vol.4, pp.2351-2354, 1995.

**57.** I.J. Cox, J. Kilian, F.T. Leighton and T. Shamoon, *Secure Spread Spectrum Watermarking for Multimedia*. IEEE Trans. on Image Processing, Vol. 6, No.12, December 1997.

**58.** L. von Ahn and N.J. Hopper, *Public-Key Steganography*. In: Advances in Cryptology: Eurocrypt'2004 (C. Cachin and J. Camenisch, eds.), vol. 3027 of Lecture Notes in Computer Science, pp. 322–339, Springer, 2004.

**59.** M. Backes, C. Cachin, *Public-Key Steganography with Active Attacks*. IBM Research Zurich Research Laboratory, CH-8803. Ruschlikon, Switzerland, August 26, 2004.

**60.** Шеннон К. *Работы по теории информации и кибернетики.* / Пер. с англ. — М.: Иностр. литература, 1963. — 829 с.

**61.** P. Moulin, J.A. O'Sullivan, *Information-Theoretic Analysis of Information Hiding*. IEEE Transactions on Information Theory, vol.49, № 3, pp. 563-593, March 2003.

**62.** J.K. Su, J.J. Eggers, B. Girod, *Analysis of Digital Watermarks Subjected to Optimum Linear Filtering and Additive Noise* // Signal Processing. Special Issue on Information Theoretic Issues in Digital Watermarking, vol.81. № 6, pp. 1141-1175, 2001.

**63.** F. Petitcolas, R.J. Anderson, M.G. Kuhn, *Information Hiding — A Survey* // Proceedings IEEE, Special Issue on Identification and Protection of Multimedia Information, vol.87, № 7. pp. 1069-1078, 1999.

**64.** F. Hartung, M. Kutter, *Multimedia Watermarking Techniques* // Proceedings IEEE, Special Issue on Identification and Protection of Multimedia Information, vol.87, № 7, pp. 1079-1107, 1999.

**65.** Б. Скляр, Цифровая связь: Теоретические основы и практическое применение. Изд. 2-е, исправл. — М.: Вильямс, 2003. — 1104 с.

**66.** M.D. Swanson, M. Kobayahi, A.H. Tewfik, *Multimedia Data-Embedding and Watermarking Strategies*. // Proceeding of IEEE, vol. 86, № 6, pp. 1064-1087, 1998.

**67.** T. Basar and G.J. Olsder, *Dynamic Noncooperative Game Theory* // SIAM Classics in Applied Mathematics. Philadelphia, PA: SIAM, 1999.

**68.** T.M. Cover and J.A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

**69.** S.I. Gel'fand and M.S. Pinsker, *Coding For Channel With Random Parameters*, Probl. Contr. Inform. Theory, vol.9, №1, pp.19–31, 1980.

**70.** Игнатов В.А. *Теория информации и передачи сигналов*. — М.: Радио и связь, 1991. — 280 с.

**71.** A.D. Wyner, *The Wire-tap Channel*. // Bell System Tech. J. 1975. Vol. 54. № 8. pp.1355-1387.

**72.** A.D. Wyner and J. Ziv, *The Rate-Distortion Function For Source Coding With Side Information At The Decoder*, IEEE Trans. Inform. Theory, vol. IT–22, pp. 1–10, Jan. 1976.

**73.** Яковлев В.А. Защита информации на основе кодового зашумления. Ч.1. Теория кодового зашумления. / Под ред. В.И. Коржика. — С.Пб.: ВАС, 1993. — 245 с.

**74.** H. Hastur, *MandelSteg*, ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/.

**75.** Шиффман Х.Р. *Ощущение и восприятие*. Изд. 5-е. — СПб.: Питер, 2003. — 928 с.

**76.** Максименко С.Д., Соловієнко В.О. *Загальна психологія:* Навч. посібник. — К.: МАУП, 2000. — 256 с.

**77.** A. Watson, *The Cortex Transform: Rapid Computation of Simulated Neural Images* // Computer Vision, Graphics, and Image Processing. 1987. Vol. 39. № 3. PP. 311-327.

**78.** S. Moller, A. Pfitzmann, I. Stirand, *Computer Based Steganography: How It Works And Why Therefore Any Restriction On Cryptography Are Nonsense, At Best*. // Information Hiding: First International Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.7-21.

**79.** T. Aura, *Practical Invisibility In Digital Communication*. // Information Hiding: First International Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.265-278.

**80.** A. Westfeld, A. Pfitzmann, Attacks on Steganographic Systems. Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools – and Some Lessons Learned // Proceeding of the Workshop on Information Hiding. 1999. — 16 p.

**81.** J. Fridrich, *A New Steganographic Method For Palette-Based Image*. // Proceedings of the ISBT PISP conference, Savannah, Georgia, Apr.1998, pp.285-289.

**82.** K. Matsui, K. Tanaka, *Video-steganography: How To secret Embed A Signature In A Picture*. // IMA intellectual properly project proceeding, vol.1, №1, 1994, pp.187-205.

**83.** M. Kutter, F. Jordan, F. Bossen, *Digital Signature Of Color Images Using Amplitude Modulation* // Proc. of the SPIE Storage and Retrieval for Image and Video Databases V. 1997. Vol. 3022. Pp. 518-526.

**84.** J. Hernandez, F. Perez-Gonzalez, J. Rodriguez, G. Nieto, *Performance Analysis of a 2-D Multipulse Amplitude Modulation Scheme for Data Hiding and Watermarking of Still Images* // IEEE Journal on Selected Areas in Communications. 1998. Vol. 16, № 5. Pp. 510-525.

**85.** A.N. Akansu, R.A. Haddad, Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets, Academic Press Inc., New York, 1992.

**86.** J. Zhao, E. Koch, *Embedding Robust Labels into Images for Copyright Protection*. // Proceeding of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques, Munich-Vienna, Verlag, Aug.1995, pp.242-251.

**87.** E. Koch, J. Zhao, *Towards Robust and Hidden Image Copyright Labeling*. // IEEE Workshop on Nonlinear Signal and Image Processing, Greece, June 20-22, 1995. Pp.123-132.

**88.** J. Smith, B. Comiskey, *Modulation and Information Hiding in Image*. // Information Hiding: First Int. Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.207-227.

**89.** I. Pitas, *A Method for Signature Casting on Digital Images*. // Int. Conference on Image Processing, vol.3, IEEE Press, Sept. 1996, pp.215-218.

**90.** M.T. Sandford, T.G. Handel, J.M. Ettinger, *Data Embedding Method*. // Proceeding of the SPIE 2615, Integration issues in large commercial media delivery systems, 1996, pp.226-259.

**91.** Коростиль Ю.М., Шелест М.Е. *Принципы построения стеганографических систем со структурной технологией*. // Труды VII международной конференции

по вопросам автоматического управления "Автоматика-2000", Львов, 11-15 сентября 2000 г., секция 7, ч.1. — Львов: ДНДІІІ. — С.273-286.

**92.** A.V. Oppenheim and R.W. Shaffer, *Discrete-Time Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ. 1989.

**93.** Л. Рабинер, Б. Гоулд, *Теория и применение цифровой обработки сигналов*. // Пер. с англ.; Под ред. Ю.И. Александрова. — М.: Мир, 1978. — 848 с.

**94.** Конахович Г.Ф., Пузиренко О.Ю. Використання пакету MathCAD v.12 для стеганографічного захисту секретних повідомлень у графічних файлах // Захист інформації: Збірник наукових праць. — НАУ, 2005.

**95.** Кошкин А., Кошкина Н. Стеганография – особенности использования программ на основе метода наименьшего значащего бита. — www.delphikingdom.ru, 2004.

**96.** Конахович Г.Ф., Пузиренко О.Ю. Використання пакету MathCAD v.12 для стеганографічного захисту секретних повідомлень в аудіофайлах // Захист інформації: Збірник наукових праць. — НАУ, 2005.

**97.** *Малая математическая энциклопедия*. // Э. Фрид, И. Пастор, И. Рейман, П. Ревес, И. Ружа. — Будапешт: Изд-во Академии наук Венгрии, 1976. — 694 с.

**98.** M. Luby, C. Rackoff, *How to Construct Pseudorandom Permutations from Pseudorandom Functions*. // SIAM Journal on Computing, 17(2): 373-386, April 1998.

**99.** Paul C. Kocher, IEEE Personal Communication, Oct. 1995.

**100.** V. Darmstaedter, J.-F. Delaigle, J.J. Quisquater, B. Macq, *Low Cost Spatial Watermarkin*g // Computers and Graphics. 1998. Vol. 5. P. 417-423.

**101.** Фильчаков П.Ф. *Справочник по высшей математике*. — К.: Наукова думка, 1974. — 744 с.

**102.** G. Langelaar, R. Lagendijk, J. Biemond, *Robust Labeling Methods for Copy Protection of Images* // Proc. of the SPIE Storage and Retrieval for Image and Video Databases V. 1997. Vol. 3022.

**103.** D. Benham, N. Memon, B.-L. Yeo, M. Yeung, *Fast Watermarking of DCT-based Compressed Images* // Proc. of the International Conference on Image Science, Systems and Technology. Las Vegas, Nevada, June 30 – July 3, 1997, vol. 1, pp. 243-252.

**104.** C.-T. Hsu, J.-L. Wu, *DCT-based Watermarking for Video* // IEEE Transactions on Consumer Electronics, Vol. 44, No. 1, Feb 1998, pp. 206-216.

**105.** Г. Корн, Т. Корн. Справочник по математике для научных работников и инженеров. Определения, теоремы, формулы. // Пер. с англ. под ред. И.Г. Арамановича. Изд. 2-е. — М.: Наука, 1970. — 720 с.

**106.** J. Fridrich, *Combining Low-Frequency and Spread Spectrum Watermarking* // Proc. of the SPIE Conference on Mathematics of Data/Image Coding, Compression and Encryption. 1998. Vol. 3456. P. 2-12.

**107.** Н. Хастингс, Дж. Пикок, *Справочник по статистическим распределениям* // Пер. с англ. А.К. Звонкина. — М.: Статистика, 1980. — 95 с.

**108.** D. Gruhl, A. Lu, W. Bender, *Echo Hiding*. Information Hiding Workshop, Cambridge, UK, (1996).

ББК 32.811.4
К338
УДК 519.688

**Конахович Г. Ф., Пузиренко О. Ю.**

К338 Комп'ютерна стеганографія. Теорія і практика.— К.: "МК-Пресс", 2006.— 288 с, іл.

ISBN 966-8806-06-9

Ця книга — одне з перших видань в галузі стеганографії. В ній викладені теоретичні та практичні основи комп'ютерної стеганографії. Представлені особливості використання сучасної системи символьної математики MathCAD v.12 з метою стеганографічного захисту інформації. Розглянуті приклади практичної реалізації приховання даних у нерухомих зображеннях, аудиосигналах і текстах.

Системно викладені проблеми надійності і стійкості довільної стеганографічної системи стосовно різних видів атак, а також оцінки пропускної здатності каналу прихованого обміну даними. Представлені результати існуючих інформаційно-теоретичних досліджень проблеми інформаційного приховання у випадку активної протидії порушника. Також розглянуті відомі стеганографічні методи, спрямовані на приховання конфіденційних даних у комп'ютерних файлах графічного, звукового і текстового форматів.

**ББК 32.811.4**

*Головний редактор: Ю. О. Шпак*

ПП Савченко Л.О., Україна, м.Київ, тел./ф.: (044) 517-73-77; e-mail: info@mk-press.com.
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавників, виготівників та розповсюджувачів видавничої продукції: серія ДК №51582 від 28.11.2003 р.

Надруковано в ПП "КОРВІН ПРЕСС". м. Київ, вул. Пшенична, 2

ISBN 966-8806-06-9

# КОМПЬЮТЕРНАЯ СТЕГАНОГРАФИЯ

**ТЕОРИЯ И ПРАКТИКА**


**Георгий Филимонович Конахович**

Доктор технических наук, профессор, декан факультета телекоммуникаций и защиты информации, заведующий кафедрой телекоммуникационных систем Института информационно-диагностических систем Национального авиационного университета.

Был неоднократно отмечен Департаментом специальных телекоммуникационных систем и защиты информации Службы безопасности Украины за весомый вклад в подготовку специалистов в области защиты телекоммуникационных систем.

Автор более 120 научных трудов, в том числе 12 монографий, учебников и учебных пособий. Имеет два патента и пятнадцать авторских свидетельств на изобретения.



**Александр Юрьевич Пузыренко**

Кандидат технических наук Национального авиационного университета, доцент кафедры телекоммуникационных систем Института информационно-диагностических систем НАУ. Автор ряда работ, по информации, опубликованных в специализированных изданиях.


Эта книга - одно из первых изданий в области стеганографии. В ней изложены теоретические и практические основы компьютерной стеганографии. Представлены особенности использования современной системы символьной математики MathCAD v.12 в целях стеганографической защиты информации. Рассмотрены примеры практической реализации скрытия данных в неподвижных изображениях, аудиосигналах и текстах.

Системно изложены проблемы надежности и стойкости произвольной стеганографической системы по отношению к различным видам атак, а также оценки пропускной способности канала скрытого обмена данными. Представлены результаты существующих информационно-теоретических исследований проблемы информационного скрытия в случае активного противодействия нарушителя. В книге изложены известные стеганографические методы, направленные на скрытие конфиденциальных данных в компьютерных файлах графического, звукового и текстового форматов.


ISBN 966-8806-06-9

Пресс

WWW.MK-PRESS.COM