# Современная стеганография. Принципы, основные носители и методы противодействия.

Шмаев Виктор Борисович
15 апреля 2005 года

Эссе по курсу "Защита информации", кафедра радиотехники, Московский физико-технический институт (ГУ МФТИ), http://www.re.mipt.ru/infsec

## 1. Что же такое стеганография?

Понятие Стеганография происходит от двух греческих слов *стегонос* (тайна) и *графи* (запись), так что слово “стеганография” можно расшифровать как “скрытое письмо” или “тайнопись”.
Основной задачей стеганографии является не шифрование и защита информации, как это делается в криптографии, а сокрытие самого факта передачи информации и существования секретного сообщения.

Стеганографию использовали ещё в V веке до н.э. Когда Гистию, находящемуся в Сузах под надзором царя Дария, нужно было послать секретное послание своему родственнику в анатолийский город Милет, он обрил наголо своего раба, и сделал татуировку с посланием на его голове. Когда волосы отросли, раба отправили в Милет.

С тех давних пор стеганография шагнула далеко вперёд. Современные криптографы применяют невидимые чернила, видимые только после специальной обработки (многие наверно в детстве и сами пользовались апельсиновым соком в качестве таких чернил, чтобы их проявить, необходимо было нагреть листок с таким посланием), микропленки, условное расположение знаков в письме и многие другие способы.

С появлением компьютерной стеганографии появились новые ещё более изощрённые возможности для сокрытия информации. Сейчас любые данные могут быть спрятаны в текст, изображение, звуковой или видео файл.

Основным требованием является избыточность контейнера для передачи информации.

## 2. Рассмотрим основные носители и способы включения нашего секретного сообщения в эти носители.

- *Текст.*

Использование стеганографии для передачи информации посредством текстовых данных достаточно затруднительно. Реализовать это можно двумя способами (хотя идея одна для обоих случаев):

1. Использовать регистр букв.
2. Использовать пробелы.

Для первого варианта, процесс заключается в следующем: пусть нам необходимо спрятать букву "A" в тексте "stenography". Для этого берем двоичное представление кода символа "A" - "01000001". Пусть для обозначения бита содержащего единицу используется символ нижнего регистра, а для нуля - верхнего. Тогда после наложения маски "01000001" на текст "stenography", результат будет "sTenograPhy". Окончание "phy" нами не использовано, поскольку для сокрытия одного символа используется 8 байт (по биту на каждый символ), а длина строки 11 символов, вот и получилось, что последние 3 символа "лишние". Используя такую технологию можно спрятать в текст, длиной N, сообщение из N/8 символов. Поскольку данное решение нельзя назвать очень удачным, часто используется технология передачи данных через пробелы. Дело в том, что пробел обозначается символом с кодом 32. Но в тексте его можно заменить также символом, имеющим код 255 или TAB'ом, на худой конец. Также как и в прошлом примере, передаем биты шифруемого сообщения, используя обычный текст. Но на этот раз 1 - это пробел, а 0 - это пробел с кодом 255.
Аналогичным образом для кодирования используется символ переноса каретки.

Как вы могли убедиться, сокрытие информации в текстовых документах не надежно, поскольку может быть легко замечено. Поэтому используются другие, более совершенные технологии...

- *Графические файлы*

Гораздо надёжнее прятать текст в изображении. Здесь всё происходит по принципу замены цвета в изображении, на близкий к нему. Программа заменяет некоторые пиксели, положение которых вычисляет сама. Этот подход весьма неплох, потому что определить технологию скрытия текста сложнее, чем в прошлом примере. Этот подход работает не только с текстовой информацией, но и с изображениями. Это значит, что можно без особых проблем в изображение tescha.jpg поместить pentagonmap.jpg, естественно если этого позволяют их размеры.

Технология использования изображений в качестве контейнера предоставляет намного более широкие возможности, нежели текстовые документы. При использовании графических форматов появляется возможность сокрытия не только текстовых сообщений, но и других изображений и файлов. Единственным условием является то, что объем спрятанного рисунка, не должен превышать размер изображения-хранилища. Для данных целей каждая программа использует свою технологию, но все они сводится к замене определенных пикселей в изображении.

- *Звуковые файлы*

Ещё более красивым решением является использование аудио форматов. Это обусловлено тем, что большинству людей даже в голову не придет, что музыка может содержать скрытую информацию. Для размещения сообщения/файла в формате MP3, используют избыточную информацию, наличие которой определяется самим форматом. При использовании других аудио файлов необходимо вносить изменения в звуковую волну, что может в очень малой степени повлиять на звучание.

## Остановимся подробнее на графических файлах, как наиболее распространённых на данный момент.

Быстрое развитие алгоритмов компрессии изображений привело к изменению представлений о самой технике внедрения секретной информации. Если первоначально предлагалось включать информацию в наименьшие значащие биты (НЗБ или LSB) для уменьшения заметности для стороннего наблюдателя, то современный подход заключается во встраивании информации в самые существенные области изображений, разрушение которых приводит к полной деградации самого изображения. Именно поэтому сейчас стегоалгоритмы учитывают свойства системы человеческого зрения, так же как алгоритмы сжатия изображений. В стегоалгоритмах часто используются те же самые преобразования, что и в современных алгоритмах сжатия (например, дискретное косинусное преобразование в JPEG формате, вейвлет-преобразование в JPEG2000).

**Итак, метод НЗБ(LSB) (Наименьшего Значащего Бита)**
Это один из самых простых методов цифровой стеганографии.
Метод использует скрытие данных с искажением контейнера, основанное на особенностях человеческого восприятия. Идея метода заключается в следующем: если взять картинку в формате BMP (или, скажем, как альтернативу 8 и более - битный оцифрованный звук), лучше всего TrueColor, в 24-х битном формате и взять да изменить младшие значащие биты цвета, то на глаз это будет незаметно. Почему именно 24 бита? Существует такой немаловажный фактор, как

объём контейнера - сколько всего можно впихнуть в картинку, пока это не станет явным. Логично предположить, что чем больше контейнер, тем больше можно и внедрить, а распространенные на сегодня 24-х битные BMP файлы - самая благодатная почва.

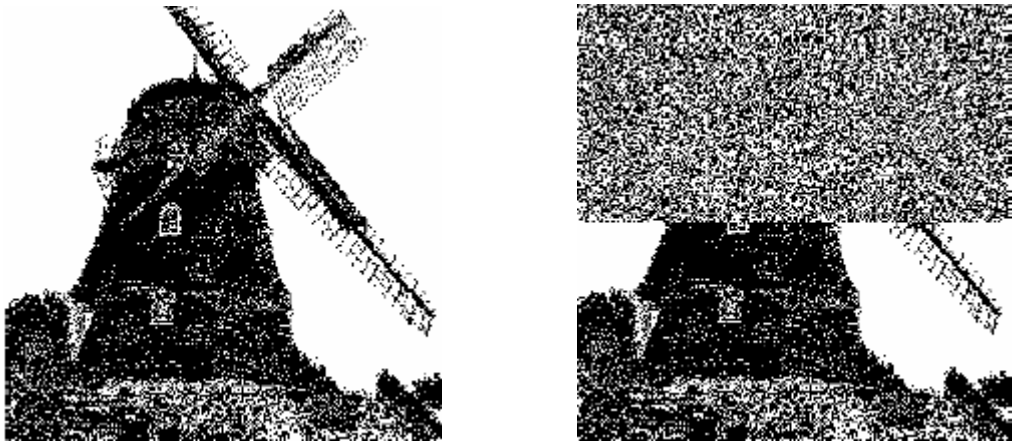
Добавление секретного сообщение можно выполнять так:

- Берём сообщение, предварительно подготавливаем его: шифруем и архивируем. Этим достигается сразу две цели - повышение КПД и увеличение стойкости системы.
- Далее берём контейнер и внедряем обработанное в первом пункте сообщение в младшие его биты любым удобным способом. Самое простое:
  раскладываем упакованное сообщение в битовую последовательность;
  заменяем избыточные биты контейнера битами сообщения.

  Вот в этом месте большая часть существующих программ и допускают большую ошибку. Надёжность подобного внедрения сильно зависит от характера распределения Наименьшего Значащего Бита в контейнере и в сообщении.
  И в подавляющем большинстве случаев эти распределения оказываются разными.
  А на картинках, построенных из одних младших битов, такое внедрение будет заметно даже на глаз (поэтому придётся ответственней подойти к выбору самого контейнера).

## Визуальная Атака

Возьмём картинку, добавим в неё сообщение с помощью программы (например, EzStego или Steganograph) и посмотрим, что будет, если посмотреть только на отдельные значащие биты:



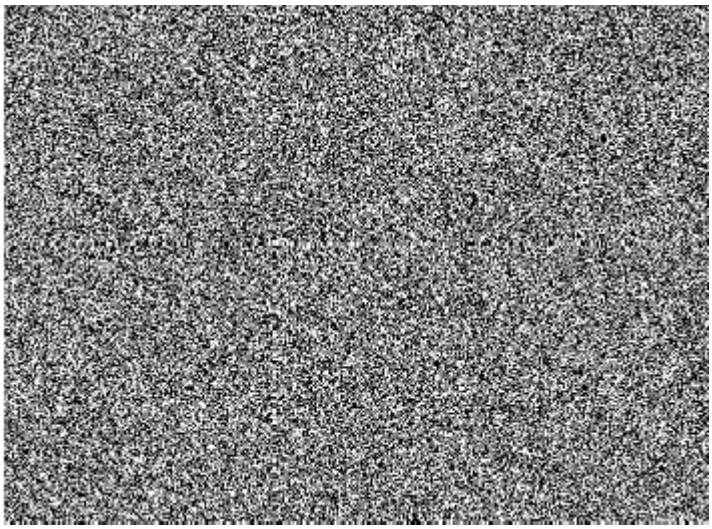
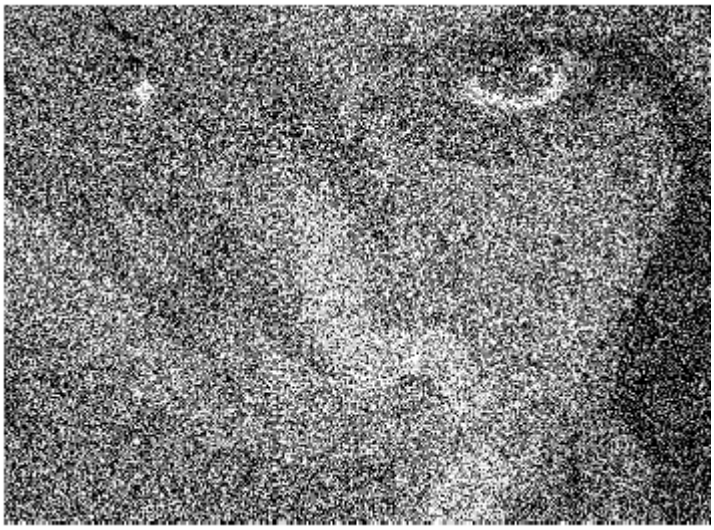
Слева картинка не подвергалась никаким преобразованиям, справа же – 50% изображения используется в качестве носителя для скрытой информации.

Так работает широкоизвестная утилита **EZStego(основана на методе последовательного внедрения - continuous embedding)**. Здесь сообщение, не использующее максимально возможный объём для скрытой информации, оставляет часть контейнера нетронутым. И достаточно легко определить, в какой именно части контейнера находится сообщение.

Утилита **S-Tools(метод распределённого внедрения - spread embedding)** распределяет сообщение на весь контейнер. В отличии от предыдущей программы, тут уже нет чёткой границы между частью изображения, не тронутой при внедрении сообщения, и частью, изменённой стеганографическими алгоритмами.


*Изначальное изображение слева, и отфильтрованные изучаемые биты этого изображения справа.*

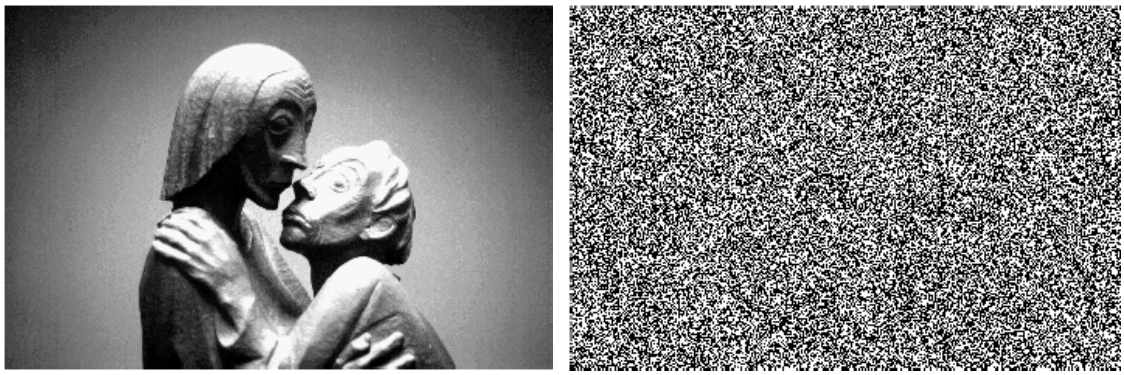
Ниже на рисунках показан результат внедрения максимально возможного размера секретного сообщения (слева) и 50% такого сообщения (справа).

*Максимальный размер* *50% внедрение*

Утилита **Steganos (метод внедрения с заполнением - continuous embedding with fill up)** всегда использует контейнер полностью. Она дополняет короткие сообщения до нужного размера. Отфильтрованные биты обработанного изображения никогда не содержат следов изначального изображения.


*Изначальное изображение* *Его отфильтрованные биты*



*Стеганограмма с одним байтом секретного сообщения (изображение и отфильтрованная часть)*

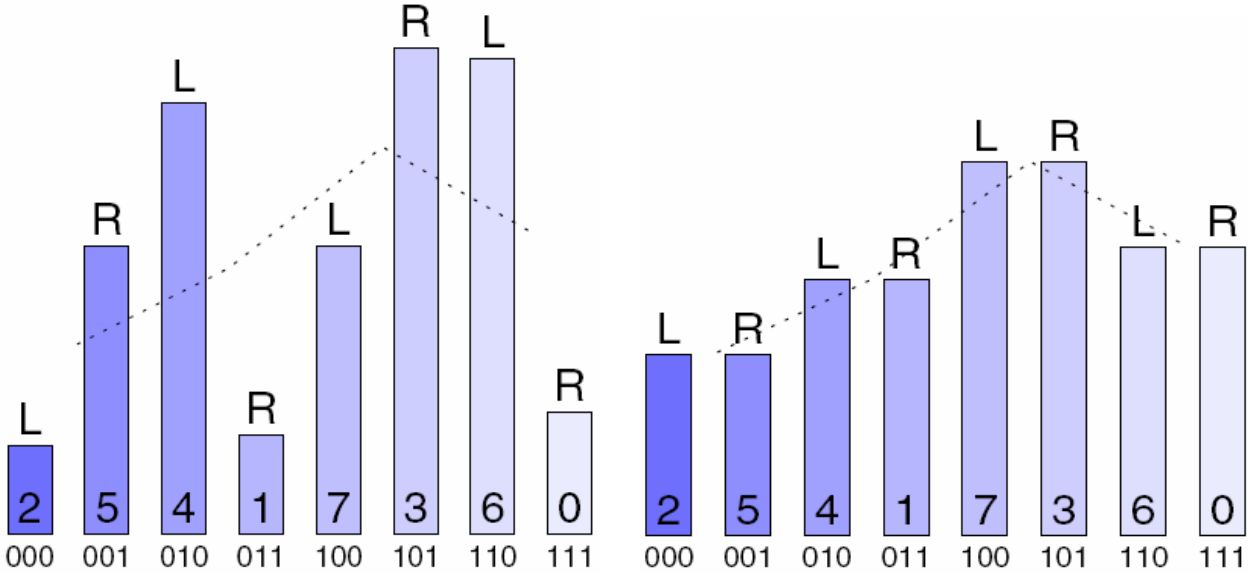
Утилита **JSTEG** включает информацию в JPEG изображения. На этот способ внедрения вышеописанная атака с отфильтровыванием нужных бит изображения не действует, так как один бит стеганограммы влияет сразу на 256 пикселов.

Прежде чем мы перейдём к следующему методу атаки, одно замечание по поводу контейнеров. Стоит в качестве таковых использовать изображения, полученные со сканеров или цифровых фотоаппаратов, так как эти приборы имеют собственные заметные шумы матриц. Вследствие чего в младших битах появится такой же примерно шум, как на рисунках выше. И на вид отличить исходное изображение от изображения с цифровика или сканера отличить будет практически невозможно.

## Статистическая атака.

Встраивающая функция EZStego переписывает наименьшие значащие биты. Соответственно меняется распределение по частоте встречи битов. Идея статистической атаки состоит в том, что сравнивается теоретически ожидаемое распределение для стеганограммы с несколькими измеренными распределениями в изображении, которое “подозревается” в том, что в него внедрено секретное сообщение. Здесь самым трудным является получить это теоретическое распределение характерное для стеганограммы (то есть частоты, которые мы ожидаем увидеть после применения алгоритмов стеганографии). В идеале нам требуется оригинал изображения, для того, чтобы сравнить виды распределения битов. Но в большинстве случаев у нас оригинала нет.

В оригинальном изображении теоретически ожидаемая частота бит – это среднее арифметическое двух частот цветов с разным последним битом (эти два цвета называются PoV). Пунктирная линия на рисунке соединяет такие средние значения.



Так как встраивающая функция переписывает наименьшие значащие биты, она не изменяет сумму этих двух частот (Частота четного бита переходит в частоту нечётного бита и наоборот). И так как сумма остаётся постоянной, среднее арифметическое будет одинаковым для контейнера и любой стеганограммы. Это позволяет нам получить теоретически ожидаемое распределение из любого примера. То есть нам не нужен оригинал контейнера для того, чтобы совершить атаку на стеганограмму.

Степень схожести наблюдаемого распределения и ожидаемого теоретически и будет мерой вероятности того, что в изображение была встроена информация. Степень схожести определяется с помощью проверки по критерию "хи-квадрат" (**Chi-square Attack**) (Проверка основана на разделение наблюдаемых распределений на категории.).

Эта проверка состоит из следующих шагов:

1) Мы предполагаем, что есть k категорий и некоторое произвольное измеренное распределение. Каждое наблюдаемое распределение должно попасть в одну и только одну категорию. Этими категориями являются индексы палитры, соответствующие цветам, расположенным на чётных местах в парах POV атакуемого изображения.

2) Теоретически ожидаемая частота в i-ой категории после встраивания сообщения с равновероятно распределёнными битами равна:

$$n_i^* = \frac{|\{\text{colour}|\text{sortedIndexOf}(\text{colour}) \in \{2i, 2i+1\}\}|}{2}$$

3) Измеренная частота в нашем произвольном примере равна:

$$n_i = |\{\text{colour}|\text{sortedIndexOf}(\text{colour}) = 2i\}|$$

4) Вводится $\chi^2$ как $\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*}$
С k-1 степенью свободы.

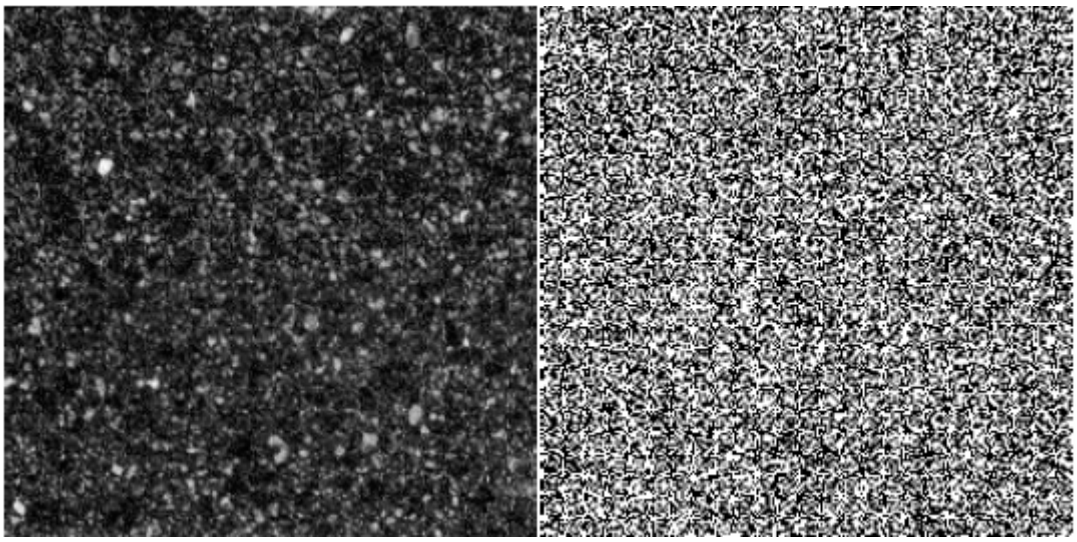
5) P это наша искомая вероятность.
Она вычисляется по следующей формуле( интергирование функции вероятности)

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} \mathrm{e}^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} \mathrm{d}x$$

Посмотрим, как работает такой метод атаки.

**EzStego**:

Опять возьмём пример с 50% встраиванием, но в этот раз с другим контейнером.



*Слева картинка со встроенной стеганограммой, справа её отфильтрованные биты.*

Теперь заметить на глаз отличии между верхней и нижней половиной картинки справа (верхняя часть – стеганограмма, нижняя – изначальные шумы) невозможно.Воспользуемся статистической атакой.

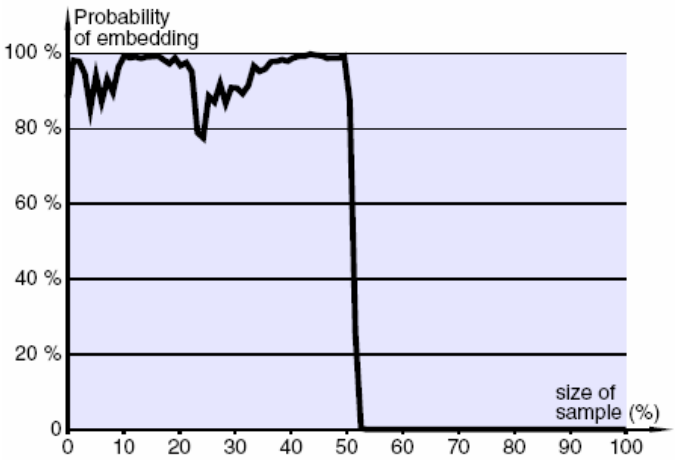


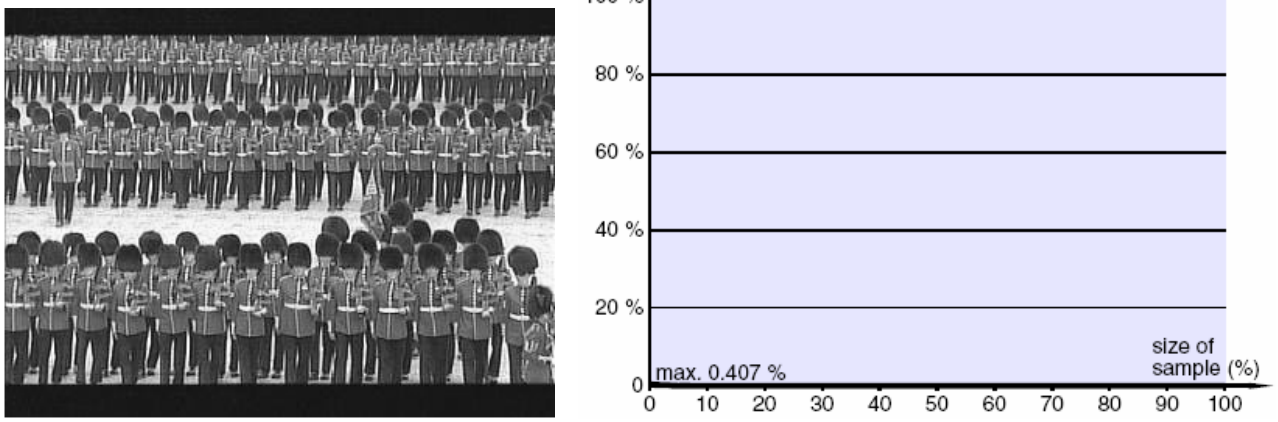

Диаграмма на рисунке представляет значение P теста “хи-квадрат” как функцию от местоположения в файле изображения. Здесь значение P означает возможное внедрение информации. Как видно из этого графика, в первой половине изображения произошло встраивание стеганограммы.

**S-Tools:**

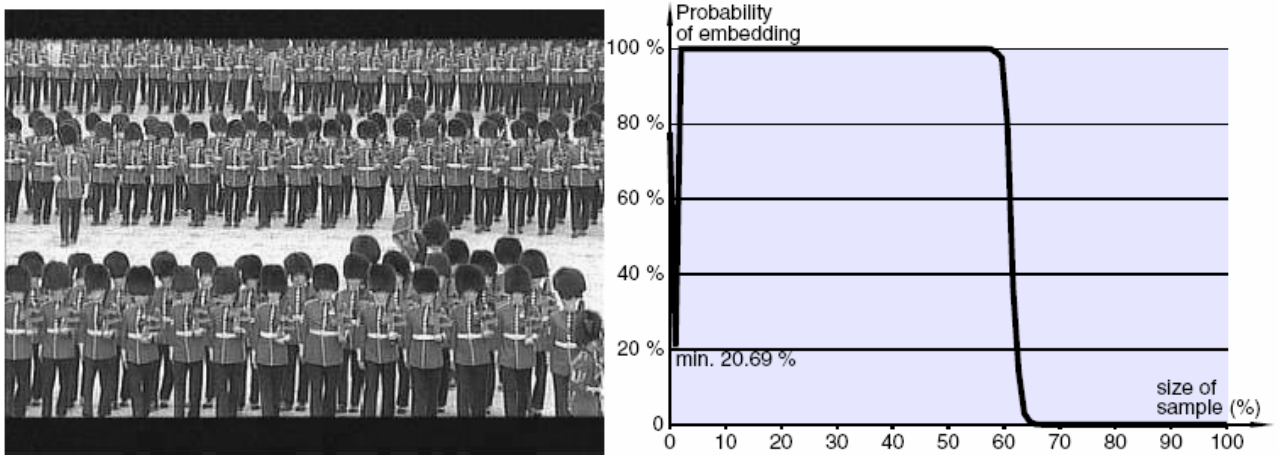
Как уже говорилось выше, эта утилита распределяет встраиваемые биты по всему контейнеру. Поэтому такие диаграммы, как в предыдущем примере не позволяют с уверенностью сказать, произошло встраивание или нет. Для стеганограмм с объемом встраиваемого сообщения около 99% этот метод может сгодиться, но если объём сообщения меньше, скажем 50% или меньше, то метод не сработает. (Существуют модификации этого метода позволяющие определять наличие 30% встраиваний.)

**Jsteg:**

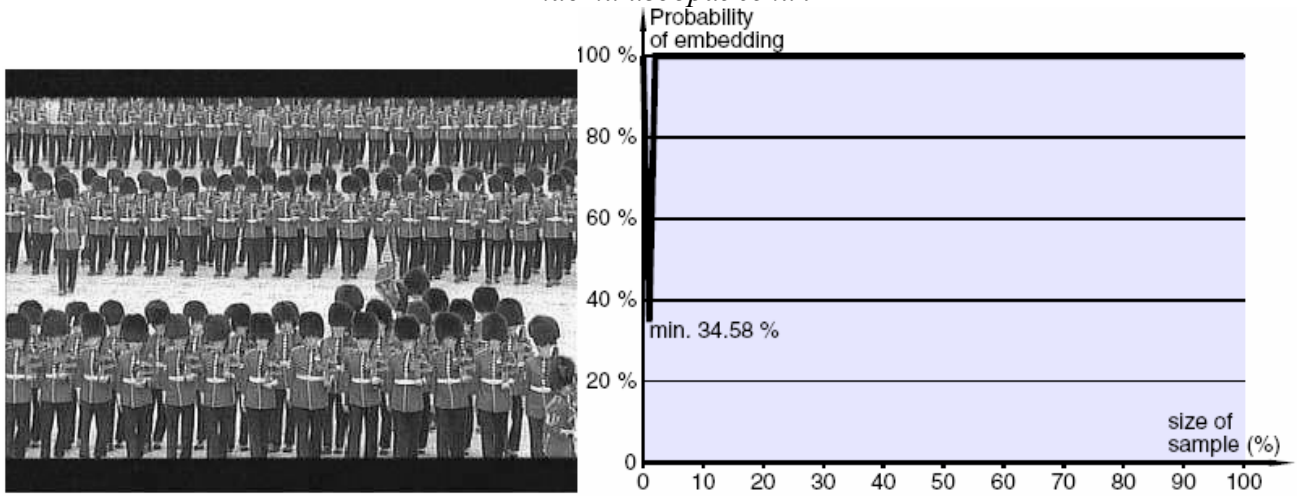
Как уже было сказано, визуальные атаки не срабатывают с этим алгоритмом встраивания. Но так как, этот алгоритм использует последовательное встраивание информации в контейнер, мы можем попробовать использовать наш статистический метод. И оказывается, что он довольно эффективен против Jsteg.




*Пустой контейнер, ничего не встроено, график показывает очень низкую вероятность наличия сообщения в картинке*




*Стеганограмма с 50% встраиванием, как видно метод показывает наличие сообщения в первой части изображения*




*И, наконец, 100% встраивание, график показывает наличие сообщения во всём контейнере*

Итак, мы рассмотрели два основных метода атаки на стеганографию, использующую в качестве носителя графические форматы файлов. На практике обычно используется статистический метод с теми или иными вариациями. Для атаки на стеганограммы построенные на звуковых файлах применяются схожие алгоритмы. Различиесостоит в том, что в звуковых файлах(в простейших алгоритмах встраивания) незначительные изменения вносятся в амплитуду звукового сигнала. Такие манипуляции достаточно легко отслеживаются статистическими алгоритмами. Современные методы стегонографии используют более сложные алгоритмы,

основанные на незначительном расширении спектра звукового файла (такой метод используется чаще остальных) и других методах. Подробное их описание потребует гораздо большего размера статьи, а он, к сожалению, жёстко ограничен. Заинтересовавшихся в этой новой и быстроразвивающейся области защиты информации я отсылаю к списку литературы, приведённом ниже.

Список литературы:


1) “Attacks on Steganographic Systems” by Andreas Westfeld and Andreas Pfitzmann. (http://os.inf.tu-dresden.de/~westfeld/publikationen/ihw99.pdf)
2) “Defending Against Statistical Steganalysis” by Niels Provos (Center of Information Technology Intergation, University of Michigan) (http://www.usenix.org/events/sec01/full_papers/provos/provos.pdf)
3) “Стеганография. Особенности использования программ на основе метода наименьшего значащего бита”. Алексей Кошкин, Наталья Кошкина (http://zeus.sai.msu.ru:7000/security/articles/min_bit/)
4) “Стеганография — 'вшивание' информации в растровые рисунки.” Юрий Писарев (http://www.delphikingdom.com/asp/viewitem.asp?catalogid=423&mode=print)
5) “Компьютерная стеганография – защита информации или инструмент преступления?” Владимир Голубев (http://www.crime-research.ru/library/Steganos.htm)
6) “Detecting Steganographic Messages in Digital Images” Hany Farid, Department of Computer Science, Dartmouth College Science, Hanover (http://www.cs.dartmouth.edu/~farid/publications/tr01.pdf)