LXF136:Стеганография

Подписка на печатную версию
Весь 2015 год (12 номеров)
Первое полугодие (6 номеров)
Второе полугодие (6 номеров)

Подписка на электронную версию
Весь 2015 год (12 номеров)
Первое полугодие (6 номеров)
Второе полугодие (6 номеров)

Подшивки старых номеров журнала (печатные версии)
Весь 2014 год (12 номеров)
Первое полугодие (6 номеров)
Второе полугодие (6 номеров)

купить
Купить электронную версию

подписаться
Подписаться на бумажную версию

Стеганография Скрываем ценные данные в обычных на вид изображениях

Содержание

1 *Perl*: Водяные знаки
1.1 Двоичные водяные знаки
1.2 Метод битовых плоскостей
1.3 Структурный способ
1.4 Голографический подход

Perl: Водяные знаки

Perl Марко Фиоретти
- Часть 1: Основы языка
- Часть 2: Регулярные выражения
- Часть 3: Контроль за выполнением
- Часть 4: Функции, модули и прочие штучки

Perl Михаил Смирнов
- *Perl*: Создаем изображения
- *Perl*: Повысим качество фото
- ***Perl*: Водяные знаки**
- *Perl*: Обнаружим объекты

Часть 3: Тот, кто придумал выражение «вилами на воде писано», явно не знал про стеганографию и не читал статьи Михаил Смирнов.

В эпоху глобальных компьютерных систем необходимость защиты конфиденциальной и секретной информации очевидна и злободневна. Во многих приложениях – например, электронном документообороте – наряду с шифрованием информации возникает потребность сделать незаметным сам факт передачи или хранения данных. Актуальность таких задач напрямую связана с ростом конкурентной борьбы, промышленным шпионажем, возрастающим контролем государственных структур над электронными средствами связи, проникновением хакеров в базы данных. Одним из наиболее эффективных способов противодействия такому вмешательству является сокрытие данных (стеганография) в массиве цифрового изображения.

Историческии, появление методов встраивания цифровых водяных знаков в файл изображения было обусловлено проблемами защиты интеллектуальной собственности – в частности, права владельцев на фотоснимки в Интернете. Решение этой задачи, в той или иной степени, достигается с помощью встраивания в защищаемое изображение невидимых скрытых цифровых водяных знаков или меток. Достоинством этого способа с использованием видимых знаков известны – это прежде всего простота процедуры наложения водяных знаков на защищаемое изображение и возможность сохранить водяной знак при печати.

Наибольшей популярностью пользуется встраивание водяных знаков с регулируемой прозрачностью. С одним из вариантов такого встраивания мы уже познакомились в первой статье цикла (LXF134). Удаление видимых графических знаков может вылиться в весьма сложную и трудоемкую задачу, и это следует отнести к достоинствам.

Недостатки встраивания видимых водяных знаков, как ни парадоксально, состоят в их видимости. Последнее обстоятельство широко используется в подпольной индустрии контрафактной продукции для тиражирования подделок изображений известных торговых брендов.

Противодействие подделкам можно осуществить с помощью методов встраивания скрытых цифровых водяных знаков или меток в файл изображения-носителя. В сфере защиты авторских прав основной недостаток стеганографических методов обусловлен неустойчивостью скрытых водяных знаков к различным видам преобразований защищаемого изображения. Вместе с тем, в сфере защиты конфиденциальной информации этот им пренебрежение, обеспечивающим разрушение данных при несанкционированном вмешательстве. Именно о реализации этих методов защиты конфиденциальной информации с помощью программного обеспечения на языке *Perl* и пойдет речь ниже.

Двоичные водяные знаки

Изображения, используемые для встраивания скрытых данных, будут представлять собой контейнеры (носители) для передачи или хранения секретной информации. При этом доступ для просмотра изображений может быть открытым и не вызывать подозрений. Для хранения данных в этом случае могут использоваться не только специализированные центры данных, но и обычные HTTP-серверы. Передаваемое сообщение предварительно шифруется и преобразуется в двоичную последовательность. На следующем этапе двоичная последовательность встраивается в массив данных файла изображения. Размерность встраиваемых данных зависит от способа встраивания, объема и яркостных характеристик изображения-контейнера. Попытки несанкционированного преобразования изображения-контейнера приводят к частичной или полной потере скрываемых данных.

Важнейшим элементом любой информационной системы является шифрование конфиденциальной информации. Рассматриваемая методология – не исключение: двумерные криптограммы органически встраиваются в двумерные изображения-контейнеры. Совместное использование стеганографического подхода и криптосистем обеспечивает практически непреодолимый барьер взлому. При этом стойкость шифра к взлому определяется исключительно стойкостью (длиной) секретного ключа.

Метод битовых плоскостей

Одним из наиболее известных способов сокрытия данных (водяных знаков) в цифровом изображении является метод встраивания в битовые плоскости изображения. Битовые плоскости или срезы – это просто разряды цифрового изображения, находящиеся в одних и тех же положениях в соответствующих двоичных числах, представляющих собой данные изображения. Их «веса» равны 2^M, где M = 0,1,2, … – номер позиции. Алгоритм встраивания основывается на свойствах зрительного восприятия и реализуется таким образом, чтобы внедряемые биты оставались бы незаметными при визуальном рассмотрении цифрового изображения. Обычно для выполнения этого условия данные встраиваются в битовую плоскость с наименьшим весом, равным 2^0 (младший бит). Чем меньше вес выбранной битовой плоскости для встраивания, тем меньшее влияние оказывается на качество изображения-контейнера, и тем менее заметны изменения. На рис.1 представлен пример двумерной двоичной последовательности данных для сокрытия.

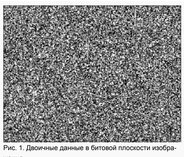

Рис. 1. Двоичные данные в битовой плоскости изображения.

Объем **Q** символов встраиваемых данных в полутоновое [grayscale] изображение можно подсчитать по формуле: **Q=P*W*H/B**, где **P** – число битовых плоскостей, используемых для встраивания, **W** и **H** – ширина и высота изображения в пикселях, соответственно, **B** — число бит на символ. На рис. 2 показаны примеры встраивания данных в битовые плоскости цветного (RGB) изображения.

Рис. 2. Изображения со встроенными двоичными данными: а) в битовую плоскость нулевого разряда и только компоненты Blue, б) в четыре битовые плоскости всех RGB-компонент.

В первом случае, на рис. 2а, для встраивания используется одна битовая плоскость наименьшего нулевого разряда 2^0 только синей (Blue) компоненты изображения. Фрагмент программной реализации для этого случая показан ниже:

```
$width3 = $width * $nbyte;
$wt = int($width3/4);
$wt = $wt*4;
$ut = $width3 - $wt;
if($ut!=0){$ut = 4 - $ut;}
for($j=0;$j<$height; $j++){
for($i=0;$i<$width; $i++){
  bin[$i][$j] = int(rand(2));
}}
$bit_plan = 1;
$rgb=0;
for($j=0;$j<$height; $j++){
  $y1=$j*($width3 + $ut);
  $k=0;
  for($i=$rgb;$i<$width3; $i+=3){
    $tmp = ($BMP_IMG[$i + $y1])<< $bit_plan|<< $bit_plan;
  $BMP_IMG[$i + $y1] = $tmp + $bin[$k][$j];
  $k++;
  }
}
```

Исходное изображение-контейнер шириной **$Width** и высотой **$Height** пикселей содержится в массиве **@BMP_IMG**. В массиве **@bin** создается случайная последовательность двоичных чисел [0,1], имитирующих шифрованную последовательность скрываемых данных. Параметр **$nbyte** равен 3 для цветного RGB-изображения. В переменной **$wt** задается количество слов в строке изображения формата **BITMAP**. В переменную **$tmp** записывается результат обнуления битовой плоскости 2^0 синей компоненты (**$rgb=0**).

Во втором примере, на рис.2б, для встраивания используются четыре битовых плоскости младших разрядов каждой из RGB-компонент соответственно. Из сравнения изображений на рис. 2а и рис. 2б видно, что чем больше битовых плоскостей занимают встраиваемые данные, тем выше степень искажений, видимых глазу. Причем, кроме отдельных шумовых выбросов, появляются ложные контуры на участках изображения с равномерным фоном (например, на голубом небе). Фрагмент программы для этого случая показан ниже.

```
$bit_plan=-;
@BMP_IMG=map($_ >> $bit_plan, @BMP_IMG);
@BMP_IMG=map($_ << $bit_plan, @BMP_IMG);
for($plan=-;$plan<$bit_plan;$plan++){
for($rgb = -1;$rgb<$nbyte; $rgb++){
for($j=0;$j<$height; $j++){
  $y1=$j*($width3 + $ut);
  $k=0;
  for($i=$rgb;$i<$width3; $i+=3){
    $BMP_IMG[$i + $y1] ++ $bin[$k][$j]*(1<<$plan);
  $k++;
}}}}
```

Обнуление битовых плоскостей осуществляется вне основных циклов с помощью функции **map()**.

Уменьшить искажения отчасти помогает применение в качестве контейнеров текстурных изображений. Статистические характеристики текстур позволяют оптимизировать соотношение между количеством встраиваемых данных и качеством визуального восприятия. Примеры текстурного контейнера при встраивании данных в четыре битовые плоскости всех RGB-компонент показан на рис. 3.

Объем встраиваемых данных в примере на рис. 3 приблизительно составляет **Q = 1,18 МБ**. Объем исходного изображения-контейнера равен 2,4 МБ. Коэффициент использования равен около 49 %.

Рис. 3. Пример текстурного контейнера со встроенными данными в четыре битовые плоскости всех RGB-компонент.

Как можно судить из названия способа, для встраивания данных без потерь необходимо использовать графические BITMAP-форматы типа BMP, TIFF и т. п. С этим связан основной недостаток данного способа, обусловленный значительным объемом файлов изображений в формате BITMAP. Если этот фактор окажется существенным, то можно воспользоваться структурным способом, представленным ниже.

Структурный способ

Настоящий способ основывается на использовании внутренней структуры графических форматов. Последняя представляет собой некоторую организацию функциональных и информационных сегментов (полей) цифрового изображения. Именно в эти поля и встраиваются секретные данные. Объем встраиваемых данных может быть не связан с размерностью изображения-контейнера. Основное достоинство способа состоит в том, что в качестве контейнера могут использоваться практически любые структурированные данные, включая медиа-файлы, документы PDF, ZIP и др. Применение графических форматов, допускающих сжатие, позволяет значительно повысить коэффициент использования файла-контейнера.

Рассмотрим программную реализацию структурного встраивания на примере формата JPEG. Деление на сегменты в этом случае осуществляется с помощью маркеров, которые принято записывать в шестнадцатеричном виде **0xFF 0xnn**, где шестнадцатеричное число **nn** указывает на то, какую функциональную нагрузку несет тот или иной сегмент. Любой маркер начинается байтом **0xFF**. Например, значения **0xFF 0xD8**, расположенные в нулевом и первом байтах файла изображения, обозначают начало изображения (SOI) JPEG-формата. **0xFF 0xD9** соответствует признаку конца изображения (EOI). Далее представлен вариант программы для встраивания секретных данных в поле перед признаком начала информационного блока **0xFF 0xC4**:

```
open FING, "images/filein.jpg";
binmode(FING );
@FILE_ING=<FING>;
close FING;
$data=join("", @FILE_ING);
@ING = unpack("C*", $data);
$marker="\xFf\xc4";
$pos =index($data, $marker);
$message="секретное сообщение";
@C1 = unpack("C*", $message);
|@BEGIN=@ING[0..$pos - 1];
@END=@ING[$pos..scalar(@ING)];
@UNIT=(@BEGIN,@C1,@END);
$out = pack("C*", @UNIT);
open JPEG, ">images/fileout.jpg";
binmode(JPEG);
print JPEG $out;
close JPEG;
```

В приведенном примере встраивание данных осуществляется во внутренний сегмент файла формата JPEG, и поэтому программа должна отслеживать возможность появления байтов **0xFF**, преобразовывать их или удалять. Можно реализовать более простой вариант программы, не требующий анализа байтов **0xFF**, если производить встраивание данных в конец файла. Ниже представлен пример встраивания в файл архива в формате ZIP.

```
open FILE, "$PATH/example.zip";
binmode(FILE);
@FILE_ZIP=<FILE_ZIP>;
close FILE;
$zip=join("", @FILE_ZIP);
@ZIP = unpack("C*", $zip);
@UNIT=(@ZIP,0xff,0xd9);
$out = $data.pack("C*", @UNIT);
```

Завершающими байтами встроенных данных будет признак конца файла: **0xFF 0xD9**. Отметим, что объем встраиваемых данных при использовании структурного способа может быть произвольным и значительно превышать размер файла изображения-контейнера, и не создавать при этом никаких визуальных искажений. Недостатком рассмотренного подхода будет предсказуемость мест для встраивания скрываемых данных. Применение обычных web-технологий позволяет устранить этот недостаток, если мы разобьем скрываемые данные на **N** частей, встроим их в **N*L** изображений-контейнеров и разместим на **L** HTTP-серверах в Интернете. С учетом развития современных web-технологий реализовать подобное совсем не сложно.

Голографический подход

Считается, что с точки зрения устойчивости скрытых водяных знаков к различным преобразованиям (атакам) целесообразно встраивать данные в сегмент видеоданных, то есть непосредственно в картинку файла изображения. Одним из интересных способов, обеспечивающим, кроме всего, сохранение данных при печати, является голографический подход. Процесс фото-печати представляет собой каскад цифро-аналоговых преобразований, которые могут оцениваться как экстремальные атаки. Фотографическое изображение формируется (если говорить кратко) в процессе вымывания и кристаллизации галоидного серебра в фотоносителе. Визуализация и степень потемнения картинки при этом обеспечивается за счет различной плотности зерен кристаллического серебра. В цифровом изображении, пикселям, содержащим только младшие биты, будут соответствовать темные участки изображения, а пикселям, имеющим и старшие биты – светлые участки. Цифровое представление изображений имеет дело с пикселями и битами, аналогово-все фотографическое характеризуется зернами и оптической плотностью. Очевидно, что методы, подобные методам битовых плоскостей, не в состоянии обеспечить сохранение водяных знаков при печати изображения-контейнера.

В голографическом подходе, в изображение-контейнер встраивается не сам водяной знак, а его цифровая голограмма.

Формула для встраивания цифровых голограмм **h(x,y)** в изображение-контейнер **g(x,y)** имеет вид:

$$s(x,y) = g(x,y) + h(x,y),$$

Рис. 4. Встраивание образца подписи: а) контейнер со встроенной голограммой, б) результат восстановления водяного знака.

где **s(x,y)** – результирующее изображение со встроенной голограммой водяного знака **W(u,v)**. Получение голограммы осуществляется с помощью обратного преобразования Фурье

$$h(x,y) = F^{-1}\{W1(u,v)\}, (1)$$

где F^{-1} – оператор обратного преобразования Фурье, **W1(u,v) = W(u-m,v-N) + W(-u-m,-v-N)** – сумма изображений водяного знака и его зеркального отражения, **m** и **N** – смещение относительно начала координат в частотной плоскости **u,v**. Параметры **m** и **N** являются ключами голограммы. Рассмотрим фрагмент кода для вычисления голограммы:

```
for($y=0;$y<$Nh; $y++){
  for($x=0;$x<$Nh; $x++){
    $wx[$x][$y]=0;
    $wy[$x][$y]=0;
}}
for($y=0;$y<$Yh; $y++){
  for($x=0;$x<$Xw; $x++){
    $wx[$x+$M][$y+$N] = $W[$x][$y];
    $wx[$Nh  $x  $M 1][$Nh  $y  $N 1] = $W[$x][$y];
}}
&d2cdfti($wx,\$wy,1$Nh,1$Nh,1$P1);
for($y=0;$y<$Nh; $y++){
  for($x=0;$x<$Nh; $x++){
    $h[$x][$y]=$wx[$x][$y];
}}
```

Двумерное обратное преобразование Фурье выполняется с помощью подпрограммы **d2cdfti()**, код которой был представлен в предыдущей статье этого цикла. Размер квадратной матрицы изображения контейнера **g(x,y)** задается параметром **$NN**, а размеры массива **@W** водяного знака **W(u,v)** задаются переменными **$Yh** и **$Xw**. Параметры смещения **$N** и **$M** не должны превышать половины **$NN** за вычетом **$Yh** и **$Xw**, соответственно.

Основной недостаток этого способа связан с ограниченным объемом встраиваемых данных. Наиболее целесообразно применять голографический подход для сокрытия небольших изображений, восстановление которых дополняет неоптическую качества (подобно JPEG): образцов подписей, отпечатков пальцев, номера и серии паспорта и т. п. На рис. 4а представлено изображение-контейнер в формате JPEG со встроенным факсимильным образчиком подписи. Качество JPEG-изображения наивысшее – 100 %. На рис. 4б показан результат восстановления образца подписи; параметры смещения **N** и **m** составляли 118 и 120 пикселей, соответственно. Вариант для сокрытия дактилоскопического отпечатка иллюстрируется на рис. 5а. Двоичное изображение водяного знака встраивалось во все три RGB-компоненты. Качество JPEG-изображения было средним и равнялось 50 %, а сжатие по сравнению со 100 % качеством составило 75 %. На рис. 5б представлен результат восстановления образца отпечатка с параметрами смещения **N** и **m** равными 199 и 204 пикселей, соответственно.

На рис. 4б и рис. 5б восстановленные образцы водяных знаков имеют зеркальное отображение, что обусловлено появлением мнимого изображения при восстановлении голограммы Фурье. Процесс восстановления голограммы является процедурой, обратной встраиванию водяных знаков, и поэтому, в соответствии с формулой (1), потребуется выполнить прямое преобразование Фурье изображения-контейнера. Отметим, что для обнаружения и восстановления секретных данных требуется формировое знание параметров **N** и **m**, при которых создавалась голограмма.

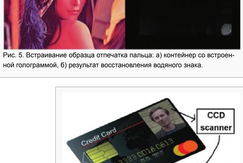

Рис. 5. Встраивание образца отпечатка пальца: а) контейнер со встроенной голограммой, б) результат восстановления водяного знака.

В случае, если изображение со встроенным водяным знаком было сохранено в виде твердой копии – например, нанесено на кредитную карту (рис. 6) – то перед восстановлением потребуется сначала отсканировать фотоизображение с карточки, а затем уже применить программное обеспечение для извлечения образца водяного знака.

Фотоизображение на рис. 6 считывается с помощью сканера на ПЗС-матрице, подвергается преобразованиям с целью восстановления скрытого водяного знака, и затем скрытая информация выводится на дисплей. На схеме, скрытые данные представляют собой графическое изображение знака копирайта ©.

Мы познакомились с весьма небольшим представительством способов для встраивания цифровых водяных знаков в файлы изображений. Вместе с тем, мы затронули важные аспекты встраивания как скрытых, так и видимых водяных знаков. Узнали, как воспользоваться облачной web-технологией при встраивании скрываемой информации во множестве изображений и размещении этих изображений на множестве серверов. Рассмотрели способ, позволяющий восстановить скрытый водяной знак из изображения, которое было предварительно напечатано на твердом носителе. По-моему, для четырех страниц – весьма неплохо.

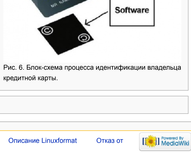

Рис. 6. Блок-схема процесса идентификации владельца кредитной карты.

Категории: Perl | Марко Фиоретти | Учебники