Journal of Information Hiding and Multimedia Signal Processing ©2012 ISSN 2073-4212
Ubiquitous International Volume **3**, Number **3**, July **2012**

A Context Sensitive Tiling System for Information Hiding

Philip C. Ritchey
Department of Computer Science
Purdue University
305 N University Drive, West Lafayette, IN, USA
pritchey@cs.purdue.edu

Vernon J. Rego
Department of Computer Science
Purdue University
305 N University Drive, West Lafayette, IN, USA
rego@cs.purdue.edu

Received February, 2012; revised July, 2012

Abstract. *We present a novel and general method to procedurally generate images for the purposes of hiding information. The generation method is realized as a two-dimensional tiling system, even though its generalization is not restricted in dimension or even in form. Empirical results on capacity, robustness and security are presented. The method can be generalized to obtain a framework that can create generic multimedia steganographic objects which hide information in a way that moves the embedding locale from the level of noise to higher levels in the semantics of the cover context. Two steganalysis methods are presented which are aimed at detecting steganography in objects generated using tiling systems. The detection methods are tested against the stego-system and their accuracies are reported.*

Keywords: Information Hiding, Steganography, Cover Generation Method, 2D Context-Sensitive Grammar, Tiling System

1. Introduction.

Steganography is the art and science of hiding secrets in innocuous-seeming objects in such a way that the existence of the secret is undetectable to a third party adversary or observer. It thus enables the covert transmission of secret information, i.e., *secret payload* or simply *payload*, between a sender and a receiver. The objects which carry the payload are known either as *cover objects* or as *stego objects*. The distinction is made based on whether the object does actually contain a steganographically concealed payload or is clean. A cover object is clean and does not contain any secret payload whereas a stego object does. The art of steganography dates back over two-thousand years [1]. The modern study of steganography grew out of concerns about *subliminal channels* — a class of covert channels that exploit digital signatures — which themselves arose as a by-product of the need for treaty verification after the Cold War [2].

Johnson and Katzenbeisser group steganographic methods into six categories: substitution systems, transform domain techniques, spread spectrum techniques, statistical methods, distortion techniques and cover generation methods [3]. Research on steganography in images has, so far, been mostly focused on augmentation of an existing image and so most methods for image stego fall into one or more of the first five categories identified by Johnson and Katzenbeisser. Few existing methods of image stego fall into the category of cover generation methods, which hide information in the structure of an object or in the steps taken in order to generate it. We now consider a number of existing image stego methods which belong to that category.

In their work on hiding information in pseudo-random images [4], Blundo and Galdi present a method for hiding information in image mosaics. First, a database of small images is maintained, and these are classified, according to their visual properties, for use in the image mosaic formation process. Next, the images are then arbitrarily separated into classes representing the binary digits one and zero. In this way it is possible to produce an image mosaic which approximates some input image and which hides secret information in the choices of sub-images that form the mosaic.

In his work on model-based steganography, Sallee approaches the generation of stego-objects from a compression point of view [5]. With access to a perfect model of some cover media, a perfect compressor can be constructed. Instances of the cover media can be given as input to the perfect compressor and it would return "perfectly compressed, truly random bit sequences". Thus, since access to a perfect compressor implies access to the corresponding decompressor, feeding any random bit sequence to the decompressor would produce a sensible instance of the cover media. This property can be used to convert a secret payload into an instance of the cover media by first processing the payload to turn it into some random bit sequence and then feeding that bit sequence to the decompressor. Alice can send the resulting stego-object to Bob without raising much, if any, suspicion. To recover the message, Bob compresses the object using the perfect compressor and reverses Alice's processing of the result. Perfect models are very hard to come by, even for simple objects. However, this method provides perfect security against any adversary with a model which is less perfect than the one used in the compression system.

Radhakrishnan, Kharrazi and Memon use a technique called *data masking* to process the entire secret message to make it appear statistically similar to some type of multimedia object [6]. According to their results, when linear and SVM classifiers were trained to detect stego in data masked images, it was found that the detectors achieved detection rates above 97%, with a false alarm rate less than 13%. Despite the weak security performance of the steganographic scheme based on data masking, the approach still seems a step in the right direction: from augmentation of existing images towards generation of novel images.

In [7] and [8], Sung, Tadiparthi, Mukkamala and Sueyoshi present techniques for hiding secret data in animations. A context free grammar is used to generate an animation wherein certain sequences of frames signify a 1 and other sequences of frames signify 0. The secret data is encoded as an animation by selecting a sequence of frames which corresponds to the next bit(s) of data to hide. It is shown that the system is secure against a passive warden.

In this paper, we explore the merits of a new type of steganographic scheme based on context-sensitive tilings. The goal is to drive payload embedding into the high level structure of an image in order to produce stego-objects with higher levels of robustness to tampering and security against steganalytic attacks. The class of images thus produced can be said to have both structure and semantics. The images are procedurally generated with the help of the secret payload and a set of rules, so that the structural content of the generated image is in itself the payload. Such a payload is easily recovered by reversing the embedding process. Here, we must emphasize that the secret information is not being hidden in an image. Instead, the secret payload is actually made to *define* an image which is unique to both payload and key.

2. **A Context Sensitive Tiling System.** In this section, we define a system for tiling a rectangular grid with uniformly-sized tiles in such a way that each tile depends, to some extent, on it's neighbors. We require uniformly-sized rectangular tiles so that any $R \times C$ grid consisting of R rows and C columns of tiles will be rectangular. The basis of our context sensitive tiling system is a 2-Dimensional Context Sensitive Grammar (2D-CSG). A 2D-CSG is a generalization of a Context Sensitive Grammar (CSG) wherein the context for each production rule includes the strings above and below as well as to left and the right of the non-terminal being replaced [9].

Definition 2.1 (Context-Sensitive Tiling System)**.** *A Context-Sensitive Tiling System T is a 4-tuple $T = (V, \Sigma, R, S)$ where V is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols disjoint with V, R is a finite set of production rules of the form*

$$\begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_4 & A & \alpha_5 \\ \alpha_6 & \alpha_7 & \alpha_8 \end{array} \rightarrow \begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_4 & \gamma & \alpha_5 \\ \alpha_6 & \alpha_7 & \alpha_8 \end{array}$$

where $A \in V$, $\alpha_i \in (V \cup \Sigma)$ and $\gamma \in (V \cup \Sigma)$, and $S \in V$ is the initial non-terminal.

A tiling system is just a 2D-CSG with a modified production rule set which limits production rules in two ways. First, only the 8 individual symbols surrounding the non-terminal under consideration may be used. Second, the non-terminal under consideration may be replaced only by a single symbol.

3. **Steganography.** A tiling system generates a grid of tiles using the set R of production rules. Because choice implies variety, and variety offers a potential for discernment, every situation that offers a rule choice simultaneously offers an opportunity to hide something meant only for a discerning receiver. With n candidate production rules, the choice can embed any combination of $\lfloor \log_2 n \rfloor$ bits. A natural order can be imposed on the production rule set by, for example, sorting them lexicographically by the right hand side. Thus, in choosing between at least 8 applicable rules, the choice of which rule to use can encode at least 3 bits, i.e., the encoding lies in the *choice*. The unused or excess rules can function as null generators, placing symbols to be ignored during the decoding/disembedding process. Or, the would-be unused rules can be shuffled in over the course of the embedding process using a shared random number generator so that every rule is equally likely to be used (conditioned on the neighbors). The shuffling of the rules corresponds to encrypting the message using a one-time pad generated from a shared random number generator. Equivalently, we can consider choices of symbols or tiles, instead of rules, as the engine of the encoding.

Algorithm 1 presents a method for generating a stego-object using a context-sensitive tiling system and a pseudo-random number generator. Alice starts in the top-left corner of the object and constructs the candidate tile set, which is the set of symbols which occur on the right-hand side of the production rules in R which are applicable to the current location. Alice then determines the number of secret message bits the location can encode, which is at most the logarithm of the size of the candidate tile set, and extracts that number of bits from the message. She then randomly permutes the candidate tile set and chooses the tile which ends up in the position indexed by the integer value of the secret message bits before moving on to the next location. This process is repeated until the message is consumed. Any remaining tiling is completed by generating random bits to use in place of the message, which is equivalent to selecting tiles at random from the candidate tile sets, until the object is completely tiled.

Algorithm 2 presents the method for decoding the stego-objects generated by Algorithm 1. Starting in the same place that Alice started, Bob also constructs the candidate tile

Algorithm 1: Stego-Object Creation.

```
Input: Input: Bit-String mssg, Tiling System TS, PRNG R
Output: Grid G
$G$ = RHS of $TS$'s initial production rule $S$
$loc$ = the initial position in grid $G$
repeat
    $T$ = CandidateTileSet($loc$,$TS$)
    $n$ = min(Capacity($loc$),$|mssg|$)
    $d = mssg[0 : n - 1]$
    $mssg = mssg[n : end]$
    $T2$ = RandomPermute($T$,$R$)
    $dn$ = BinaryToInteger($d$)
    $G[loc] = T2[dn]$
    $loc$ = next position
until $|mssg| == 0$;
foreach remaining loc in G do
    $T$ = CandidateTileSet($loc$,$TS$)
    $n$ = Capacity($loc$)
    $d$ = RandomBitString($n$)
    $T2$ = RandomPermute($T$,$R$)
    $dn$ = BinaryToInteger($d$)
    $G[loc] = T2[dn]$
    $loc$ = next position
end
```

Algorithm 2: Stego-Object Decoding.

```
Input: Grid G, Tiling System TS, PRNG R
Output: Bit-String mssg
$loc$ = the initial position in grid $G$
repeat
    $T$ = CandidateTileSet($loc$,$TS$)
    $n$ = Capacity($loc$)
    $T2$ = RandomPermute($T$,$R$)
    $dn$ = index of $G[loc]$ in $T2$
    $d$ = $n$-bit binary representation of $dn$
    $mssg$ = Concatenate($mssg$,$d$)
    $loc$ = next position
until loc not in G;
```

set, determines the number of bits that could have been encoded in this location and randomly permutes the candidate tile set (Alice and Bob share the same PRNG and seed so that they will both generate the same random permutation). Bob then determines which element of the permuted set Alice chose and records the binary value of this index as the secret message bits before moving on to the next location. Bob repeats this process until the entire object has been decoded.

3.1. **An Example of the Proposed Scheme.** Suppose Alice wishes to send a message to Bob using the above proposed scheme using the tile set shown in Figure 3 and Rule Set 1 (shown in Equation 5). Let the message she wishes to send be $m =$

88888888, a decimal number having eight eights. The decimal number 88888888 is expressed in binary as 101010011000101011000111000, which has 27 bits. Alice will send this 27 bit sequence preceded by an 8-bit content-length field that will tell Bob how many bits to read after the first 8. Thus, the full data payload that Alice will send is: 00011011101010011000101011000111000, which has length 35 bits. Alice knows from Table 1 that Rule Set 1 encodes at least 2 bits per tile and so knows that her 35-bit message will require no more than 18 tiles. Alice rounds up to the nearest square and decides her tiling will be $5 \times 5 = 25$ tiles. By convention, and to simplify this example, Alice and Bob use blank initial state tiles. The will also mirror their images horizontally and vertically. Mirroring this way makes the image more visually appealing and has the effect of making the tiling immune to rotations and reflections as well as more robust to other tampering from Wendy. Alice and Bob will use a shared key with value 12345, which is used as the seed r_0 for their pseudo-random number generators (PRNGs). For this example, Alice and Bob will use the following 32-bit linear congruential generator as their PRNG:

$$r_{n+1} = \lfloor \frac{1}{2^{16}} \left(25214903917 \cdot r_n + 11 \pmod{2^{48}}\right) \rfloor. \tag{1}$$

To begin, Alice sets out a 6×6 grid of nonterminal open symbols, which will be replaced with terminal tiles as the tiling is constructed. She replaces the top row and the first column of open symbols with blank tiles, per her shared convention with Bob. Then, she begins with the first open symbol in the top right corner and considers which candidate tiles she can place at this location given the three neighbors to the North, Northwest and West. Under Rule Set 1, given three blank neighbors, the candidate tile set contains the tiles {0,3,5,6,7}, which means Alice can hide 2 bits of secret in her choice of tile for this location. She removes these 2 bits from the message (these first 2 bits are 00) and converts the bitstring to a decimal number, in this case 0. She rolls her PRNG once and uses the result to generate a permutation of the candidate tile set, in case she obtains {0,6,3,5,7}. She selects the 0th tile from this permuted set (because the data segment she is hiding has decimal value 0) and replaces the open symbol at the current location with the selected tile. The next location is the grid cell directly adjacent to the east of the current location. Alice again considers the neighbors (all blank tiles) to obtain the candidate tile set ({0,3,5,6,7}), extracts 2 more bits of secret (bitstring is 01), uses the PRNG once to randomly permute the candidate tile set ({7,0,5,3,6}), selects the permuted candidate tile whose index is the decimal value of the secret bitstring (another 0 tile) and finally she moves to the next location. At this location, the neighbors are all 0 tiles still, the secret bitstring is 10, the permuted candidate tile set is {0,3,5,7,6} and so the tile Alice selects at this step is the 5 tile. At the next location, the neighbors are the tiles {0,0,5}, the secret bitstring is 11, the permuted candidate tile set is {15,12,9,14,13,10,11} and so the tile that Alice selects at this step is the 14 tile. She will repeat this process until she runs out of secret message to send, at which point she will start hiding random bits. The state of the tiling after each iteration is shown in Figure 1. Once she has set the last tile, she trims the initial state tiles from the edges and mirrors the image twice, once horizontally and once vertically, before sending the final image, shown in Figure 2, to Bob.

When Bob receives the image, he can easily reverse the last steps that Alice took before sending the image. He undoes the mirroring (or uses it to undo any tampering by Wendy) and adds back in the initial state tiles around the edges (he knows these because he and Alice have a prior agreement on what they are or how to generate them) to obtain an image from which he can extract the secret data. Bob starts in the same location that Alice

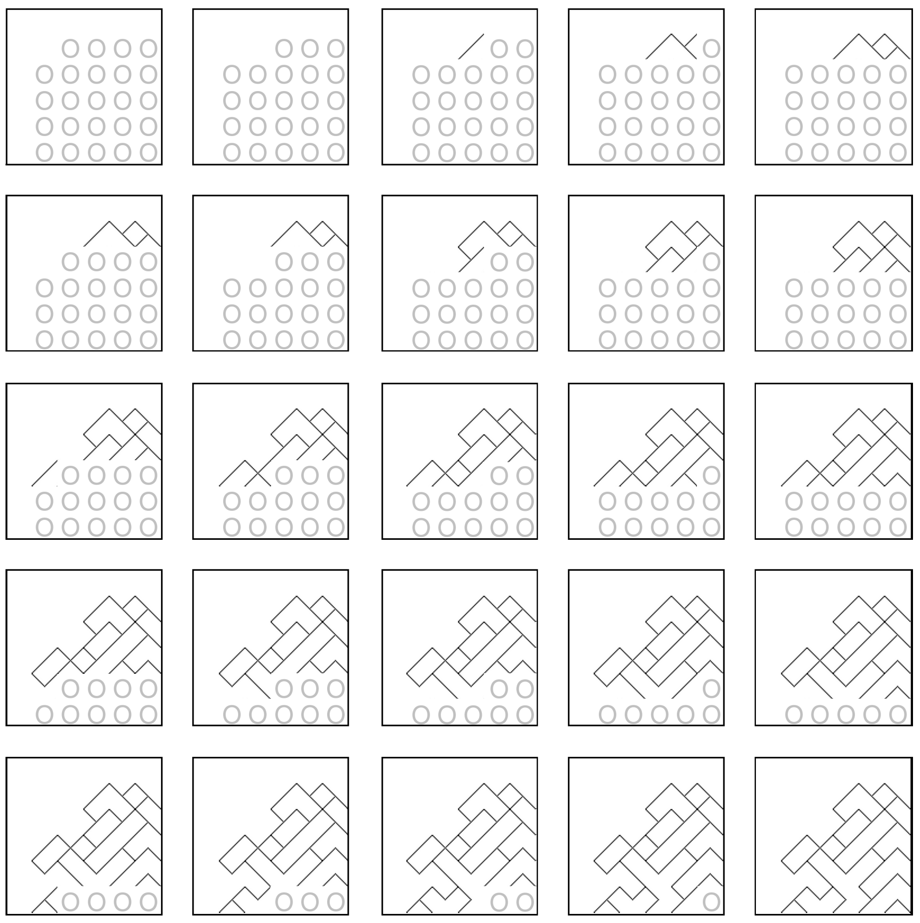


Figure 1. Snapshot of the example tiled image after each tiling iteration. The 'O' symbols, which are the nonterminal open symbols, and the border are shown for illustration purposes only.

started, but instead of an open symbol Bob sees a tile in the first location. Nonetheless, Bob uses the three neighbors to construct the candidate tile set, uses the size of the set to determine how many bits to expect, uses his PRNG and the key he shares with Alice to randomly permute the candidate tile set and then finds the index in the permuted set of the tile which he finds in the current location. In this case, the permuted tile set is {0,6,3,5,7}, the tile in the current location is 0, which is in index 0, and the number of bits to expect is 2. Therefore, Bob records the first 2 bits of the secret as `00`. In the next location, the permuted tile set is {7,0,5,3,6}, the tile in the current location is 0, which is in index 1, and the number of bits to expect is 2, so Bob records the next bits of the secret as `01`. Likewise, for the third location, the permuted tile set is {0,3,5,7,6}, the tile in the current location is 5, which is in index 2, and the number of bits to expect is 2, so Bob records the next bits of the secret as `10`. At the fourth location, Bob will find that the permuted tile set is {15,12,9,14,13,10,11}, the tile in the current location is 14, which is in index 3, and the number of bits to expect is 2, and so Bob will record the 7th and 8th bits of the secret as `11`. Bob has 8 bits now and so he can find out how many more bits to expect. The bits Bob has are `00011011`, which is the number 27 and the correct number of bits to expect to extract from the rest of the tiling. Bob will repeat the procedure to extract bits from the tiling until he has extracted the expected number of bits and then will translate the bitstring of data he has extracted to something meaningful. In this

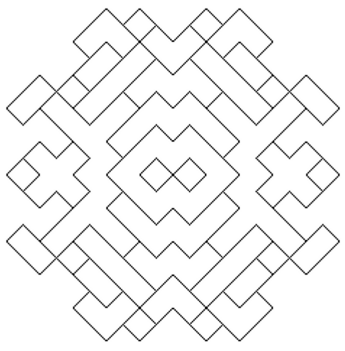


FIGURE 2. Finished line drawing encoding decimal number 88888888.

case, Bob will find out that the next 27 bits are `101010011000101011000111000`, which translates to 88888888 in decimal.

4. **Steganalysis.**

Numerous image steganalysis tools are freely available on the internet, including StegSecret [10], Virtual Steganographic Laboratory (VSL) [11] and Digital Invisible Ink Toolkit (DIIT) [12]. All three tools implement the RS Steganalysis technique [13], which classifies images based on a smoothness metric. The tools also implement other methods in addition to RS Steganalyis. StegSecret implements the Chi-Square Attack [14] and a technique the author of the tool created based on the ideas in [14] called Visual Attacks. The VSL tool implements a steganalysis method based on the Binary Similarity Metric [15]. And, the DIIT tool implements Laplace Graph Analysis [16].

However, when these methods are tested against objects generated by a tiling system, their accuracy is no better than guesswork. Clearly, one reason for this is that objects generated by tiling systems are a different type of image than these steganalysis methods are expecting. For this reason, it is believed that successful detection of the stego-objects generated by a tiling system requires an approach that draws on methods from other domains, such as linguistic steganalysis. Two steganalysis techniques specifically designed to distinguish between clean tilings and stego tilings are presented here.

4.1. **Markov Chain Equivalence Test.**

The first directed steganalysis method treats each tiled image as a sequence of steps drawn from a third order Markov chain. The neighboring tiles are the previous states and the tile under consideration is the state to which the process moved. The empirical transition matrix is constructed for each tiled image being tested. Using a method adapted from [17], the score for the image is computed as the Manhattan distance of the empirical transition matrix T from the model transition matrix for clean tiled images Tc,

$$s(T) = \sum_{x,y,z,w \in \Sigma} |Tc_{x,y,z,w} - T_{x,y,z,w}|. \tag{2}$$

The clean model is built by averaging the transition matrix over 1000 examples of clean tiled images made using a given production rule set. Allowing Wendy to know the rule set that Alice and Bob are using is not required by Kerckhoffs's law [18] and it does make Wendy's job easier. However, if Wendy watches Alice and Bob's communication long enough, she could learn the production rule set they were using. Thus, we can assume that Wendy knows the rule set from the beginning.

4.2. **Chi-Square Test.** The other directed steganalysis technique counts the occurrences of each tile in the tiled images being tested and compares those values to the model counts for a clean image using a Chi-square test with a Laplace correction. The score for each tiled image T is then

$$s(T) = \sum_{t \in \Sigma} \frac{(O_t - E_t)^2}{E_t + 1}, \tag{3}$$

where O_t is the number of t tiles observed in T and E_t is the expected number of t tiles from the model of a clean image Tc.

As before, the clean model T_c is obtained by averaging the tile counts over 1000 examples of clean tiled images. Likewise, Wendy is again assumed to know the production rule set being used.

5. **Experimental Results.** Capacity, robustness and security are three key properties of covert channels. The channel capacity is the number of bits that can be sent per transmission, typically measured in bits per object or bits per symbol. The robustness of the channel is the types and the amounts of noise or tampering that the channel can endure without preventing covert communication. The steganographic security of the channel is a measure of the indistinguishability of the steganographic usages of the channel (the stego-objects) from the non-steganographic usages (the cover objects).

Experimental results on capacity, robustness and security were obtained for an example tiling system using square tiles composed of lines connecting the center of the tile to zero or more of the corners. Four production rule sets were devised to investigate the effect of context sensitivity (the constraints placed on candidate tile sets by the production rules) on capacity, robustness and security. The example tile set is shown in Figure 3 and examples of tiled images generated using each production rules set are shown in Figure 4.

Definition 5.1 (Experimental Line Drawing Rule Sets)**.** *Let $b_x(\alpha)$ be 1 if the tile α has a line connecting the center of the tile to the corner indexed by x and 0 otherwise. Then, let the notation be abused so that $b_{\{X\}}(\{A\})$ counts the number of lines which connect each of the centers of the specified tiles $\alpha_i \in A$ to the specified corners of those tiles $x_i \in X$ (one corner for each tile), which is computed as follows.*

$$b_{\{X\}}(\{A\})) = \sum_i b_{x_i}(\alpha_i) \ . \tag{4}$$

Then, the four rule sets used for this experiment can be written as follows, where corners are numbered from 1 to 4 in clockwise order from the top-left corner and the neighbors are the adjacent tiles to the NW, N, and W of the current location.

Rule set 1:

$$\gamma \in \begin{cases} \{0,3,5,6,7\} & b_{3,4,2}(\alpha_1,\alpha_2,\alpha_3) = 0 \\ \{9:15\} & b_{3,4,2}(\alpha_1,\alpha_2,\alpha_3) = 1 \\ \{0,3,5,6,7,9:15\} & otherwise \end{cases} \tag{5}$$

Rule set 2:

$$\gamma \in \begin{cases} \{0:15\} & \alpha_1 = \alpha_2 = \alpha_3 = 0 \\ \{0:7\} & b_{3,4,2}(\alpha_1,\alpha_2,\alpha_3) = 0 \\ \{8:15\} & b_{3,4,2}(\alpha_1,\alpha_2,\alpha_3)) = 1 \\ \{0:15\} & otherwise \end{cases} \tag{6}$$

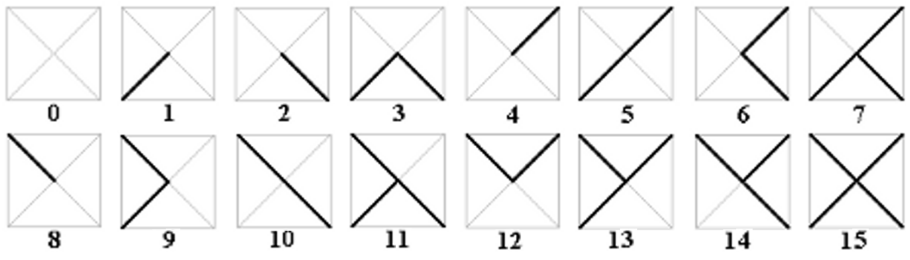
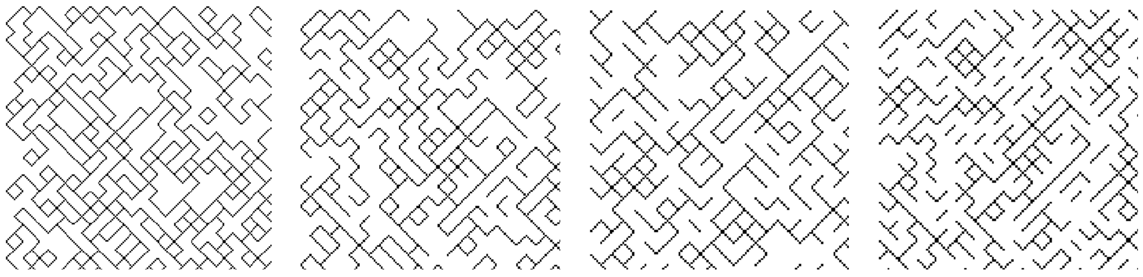


FIGURE 3. Example tile set for $N = M = 4$ as diagonal crosses.



(a) Rule Set 1 (b) Rule Set 2 (c) Rule Set 3 (d) Rule Set 4

FIGURE 4. Examples of line drawings created using each rule set.

Rule set 3:

$$\gamma \in \{0:15\} \tag{7}$$

Rule set 4:

$$\gamma \in \begin{cases} \{0:15\} & \alpha_1 = \alpha_2 = \alpha_3 = 0 \\ \{8:15\} & b_{3,4,2}(\alpha_1, \alpha_2, \alpha_3) = 0 \\ \{0:7\} & b_{3,4,2}(\alpha_1, \alpha_2, \alpha_3) = 1 \\ \{0:15\} & otherwise \end{cases} \tag{8}$$

5.1. **Capacity.** To measure the covert channel capacity of the tiling system, a square tiling of 400 tiles was generated 1000 times for each of the 4 experimental rule sets and 2 different encoding modes using random hidden messages. The two encoding modes are fixed-length encoding (FLE) and variable-length encoding (VLE). In the FLE mode, each tile choice encodes the same number of bits, equal to the minimum capacity guaranteed by the production rule set, shown in Equation 9.

$$N_F = \min_{\{x,y,z\} \in LHS(R)} \{\lfloor \log_2(|T(\{x,y,z\})|) \rfloor\}, \tag{9}$$

$T(\{x,y,z\})$ is the candidate tile set for a grid location with neighbors $\{x,y,z\}$.

In the VLE mode, each tile choice encodes a variable number of hidden message bits as shown in Equation 10. In this mode, the encoding capacity of each tile choice is just the logarithm of the size of the candidate tile set for the given location.

$$N_V(r,c) = \lfloor \log_2(|T(\{x,y,z\})|) \rfloor, \tag{10}$$

where $\{x,y,z\} = \{G(r-1,c-1), G(r-1,c), G(r,c-1)\}$ are the neighbors of $G(r,c)$ and $T(\{x,y,z\})$ is the candidate tile set for a grid location with neighbors $\{x,y,z\}$.

		Encoding Method	
		FLE	VLE
Rule Set	1	2.0000 ± 0.0000	2.5601 ± 0.0014
	2	3.0000 ± 0.0000	3.4588 ± 0.0015
	3	4.0000 ± 0.0000	4.0000 ± 0.0000
	4	3.0000 ± 0.0000	3.4603 ± 0.0015

TABLE 1. Embedding capacity results, 95% confidence intervals reported in bits/tile.

The 95% confidence intervals about the means for the capacity experiments are reported in Table 1. The results show that the VLE mode encodes fractionally more bits per tile than the FLE mode for rule sets 1, 2 and 4, about half a bit per tile. Rule set 3 is the same for both modes because it does not constrain tile choices, and thus the FLE capacity is identical to the VLE capacity in that case. Because rule set 3 puts no constraints on the relationship between neighbors, it yields the highest capacity of the 4 rule sets tested. Rule set 1 is the most constrained rule set and so has the lowest expected capacity. Rule set 2 relaxes the constraints of rule set 1 and, being less constrained than rule set 1 and more constrained than rule set 3, it's expected capacity is between those of rule set 1 and rule set 3. Rule set 4 has nearly identical capacity to that of rule set 2 because the two are constrained by the same amount.

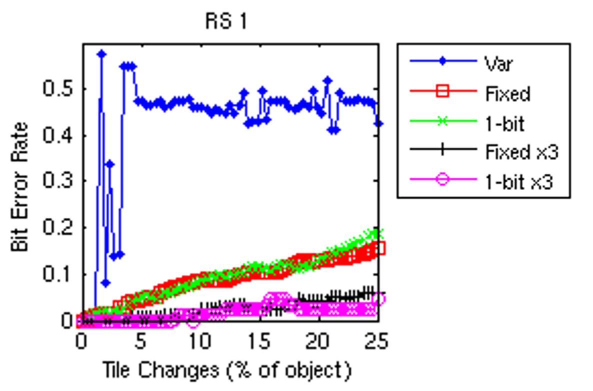
5.2. **Robustness.** The robustness of a stego-object is a measure of the degree of tampering it can tolerate without corruption of the hidden data. Because stego-objects can be subjected to various forms of tampering, it is important to be clear on precisely which forms of tampering a stego-object can withstand. In general, most tampering processes seek to augment the object with specific kinds of noise.

A popular method of tampering is the addition of white noise to a stego-object. The addition of such noise to an image or audio file can obliterate LSB stego without noticeable degradation in the quality of the object. If our stego-objects are represented as binary images, we can assume that the addition of white noise is not a viable tampering technique as it would result in clearly noticeable object degradation. On the other hand, if we represent our stego-objects as grayscale images, then the addition of white noise will be noticeable, but will not affect the secret payload.

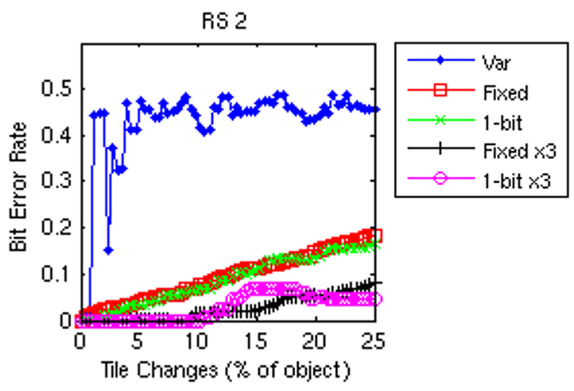
If we subject ourselves to Kerckhoffs's law so that Wendy knows the stego-system but not the key, it is reasonable to assume that Wendy may add a specific kind of noise to the image by swapping tiles with grammatically valid alternatives. To measure the robustness of objects under this operation, we allow Wendy to change as many tiles as she wants, with the constraint that she can change a specific tile location no more than a fixed number of times in total. Under this condition, we then measure the direct impact on the secret payload in number of errors in payload after each change.

To measure the robustness of the generated stego-objects, we ran one test for each of three encoding lengths (variable-, fixed-, and unit-length), and one for both fixed-length and unit-length (i.e., 1 bit/tile) encodings with redundancy. We can add redundancy by enlarging the secret message through (redundant) repetition k times. In our tests of robustness with redundancy, the payload was an ASCII encoded version of a secret message repeated $k = 3$ times. A simple majority vote is used to decide on the final value of decoded bits.

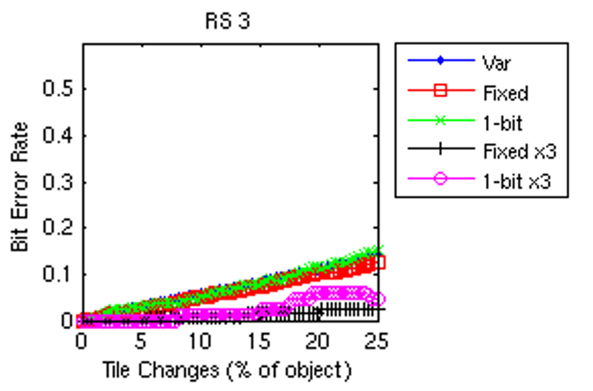
The results of the robustness experiments are shown in Figure 5. In the figure, the x-axis is the number of tile changes relative to the original object and the y-axis is the bit-error rate. It is clear from the figures that fixed length encodings exhibit a performance that is superior to that of variable length encodings, in terms of robustness. At a tampering rate



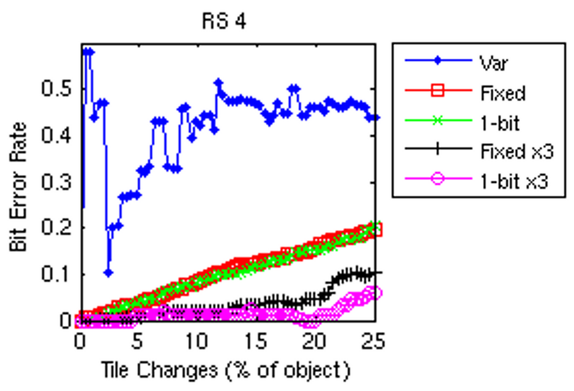
(a) Rule set 1.



(b) Rule set 2.



(c) Rule set 3.



(d) Rule set 4.

FIGURE 5. Robustness results.

of 25%, i.e. 25% of tiles changed by Wendy, variable-length encodings have a bit-error rate of close to 50%, while both fixed- and unit-length encodings have error rates less than 20%. We also observe that redundancy increases robustness to tampering: with 25% tiles changed, fixed-length encoding with redundancy achieves a bit-error rate of less than 10% and that of unit-length is less than 5%.

Variable length encodings perform poorly because tiles that previously encoded n bits under Alice's direction can be subjected to a neighbor change (by Wendy) that causes Bob's decoding algorithm to find the same tile now encoding $m \neq n$ bits, effectively shifting the rest of the message and causing numerous and simultaneous errors. The effect of such tampering could be mitigated by having a human in the decoding loop. In the case of ASCII encodings, for example, the point at which such an error is encountered will be clear because the payload prior to that point is semi-readable (possibly due to Wendy's partially successful tampering), while the subsequent payload is unreadable. It is possible for Bob to fix the message by adding or removing bits, continuing the recovery scheme until the next point is met for which too many or too few bits are decoded. Further, we believe it may be possible to automate this recovery process.

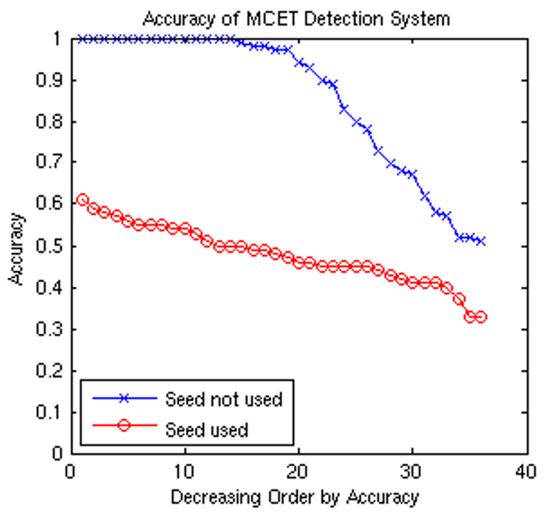
5.3. **Security.** To measure the steganographic security of the tiling system, the output of the system is tested against both directed steganalysis techniques presented above. The steganalysis testing process consists of two steps: a learning step and a testing step. During the learning step, Wendy computes the score of each tiled image in the training set and then finds the decision threshold which yields the equal error rate, the value at which the type I error rate is equal to the type II error rate. Then, this decision threshold is used in the testing step to obtain the values necessary to compute the accuracy, which is then recorded. Wendy is given a training set of 200 images, 100 clean and 100 containing hidden information, and 200 test images, again 100 clean and 100 containing hidden information.

5.4. **Markov Chain Equivalence Test.** The steganalysis test was executed for all 72 possible combinations of parameters: use of seed, choice of rule set, message format, and embedding format. The results of the tests are shown in Figure 6. The data is shown sorted in decreasing order by accuracy, holding one parameter constant. From the results, we can see that if Alice and Bob do not use a seed, Wendy can achieve nearly perfect accuracy of detection for approximately half of the parameter combinations. However, when a seed is used, Wendy does not achieve any significant advantage in detection over pure guesswork. The difference in Wendy's accuracy between when Alice and Bob use a seed and when they do not is statistically significant since the usage of a seed yields lower accuracy for Wendy with probability 98%. There is no significant difference between rule sets, in terms of Wendy's detection accuracy, nor is there a significant difference between message encoding types or between embedding types. Overall, Wendy's detection accuracy with the Markov chain equivalence test method averages 0.67 with a standard deviation of 0.23.

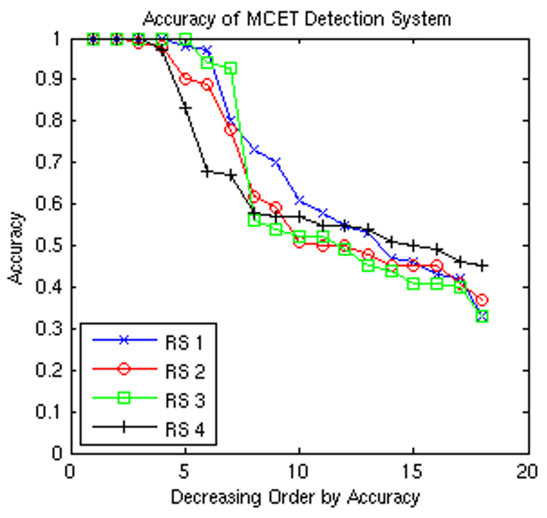
5.5. **Chi-Square Test.** The steganalysis test was again executed for all 72 possible combinations of parameters and the results are shown in Figure 7. The data is shown sorted in decreasing order by accuracy, holding one parameter constant. We again see from the results that when Alice and Bob do not use a seed, Wendy achieves perfect accuracy of detection. However, now she can do it for nearly all configurations of the other parameters. When a seed is used, Wendy's accuracy is identical to pure guesswork, with an average accuracy of 0.50. As with the Markov chain equivalence test method, this difference in Wendy's accuracy is statistically significant as Wendy's accuracy when Alice and Bob use a seed is lower than when they do not with probability 99%. We find again that there is no significant difference between rule sets in terms of Wendy's detection accuracy, nor is there a significant difference between message encoding types or between embeding types. Overall, Wendy's detection accuracy with the Chi-square test method averages 0.72 with a standard deviation of 0.25.

5.6. **Analysis of Security Results.** Taking into account only the 36 parameter combinations that yield the worst results for Wendy (which is roughly the set for which Alice and Bob are using a seed), her detection accuracy using the Markov chain equivalence test method has a mean of 0.47 with a standard deviation of 0.06. For the Chi-square test method, Wendy's detection accuracy has a mean of 0.49 with a standard deviation of 0.05. Thus, we can conclude that neither method performs any better than random chance when Alice and Bob choose one of the "better" parameter combinations, such as {useSeed = TRUE, RS = 1, ASCII printable character encoding, fixed-length embedding} for which Wendy achieves accuracy of 0.58 for the Markov chain equivalence test method and an accuracy of 0.51 for the Chi-square test method or {useSeed = TRUE, RS = 4, pseudo-random bit string encoding, 1-bit embedding} for which Wendy achieves accuracies of 0.51 and 0.43 using the respective test methods.

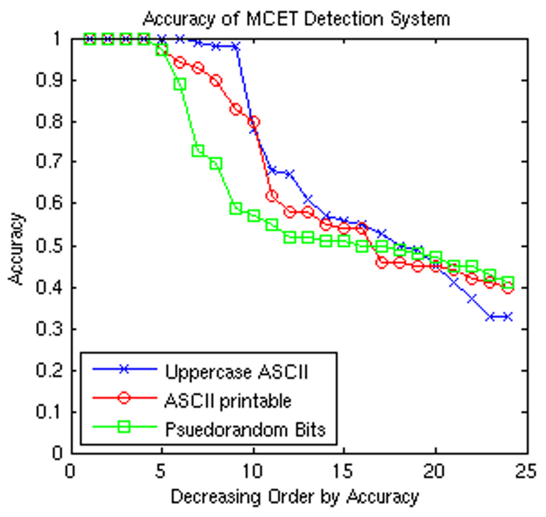
From the results obtained for the Markov chain equivalence test method and the Chi-square test method, it is found that the two methods yield quite similar results. The results also indicate that the usage of a seed significantly reduces Wendy's accuracy using these two methods. However, there is no significant difference in detection accuracy between rule sets, message encoding or embedding types.



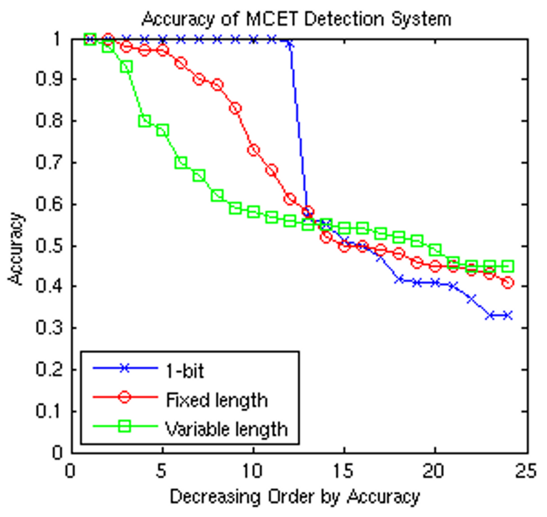
(a) Conditioned on Use of Seed



(b) Conditioned on Rule Set



(c) Conditioned on Message Type



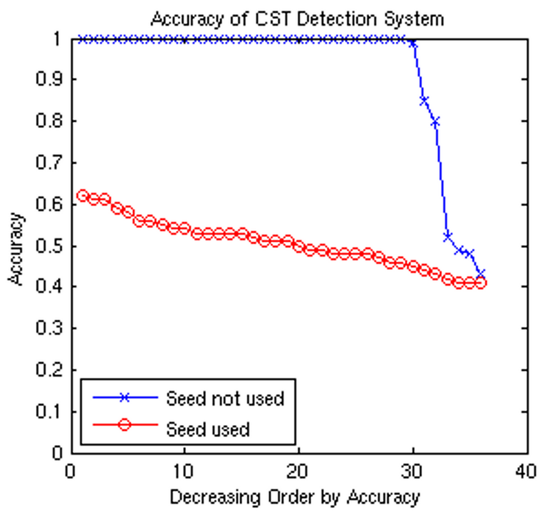
(d) Conditioned on Embedding Type

FIGURE 6. Detection accuracy results for Markov Chain Equivalence Test.

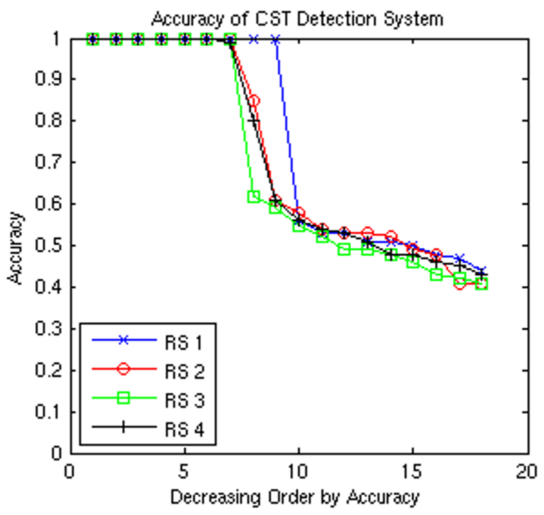
Alice and Bob can still make Wendy's detection task even more difficult by spreading the message across several images in a series, so that only part of an image encodes information and only some of the images which are sent contain any hidden information at all. This reduction in Alice and Bob's covert channel usage directly translates into an increase in the security of their channel, in terms of Wendy's detection accuracy. Further, the additional images add to the robustness of the channel so that Wendy must do more work (change more tiles) in order to cause enough damage to destroy to hidden information.

6. **Conclusion.**

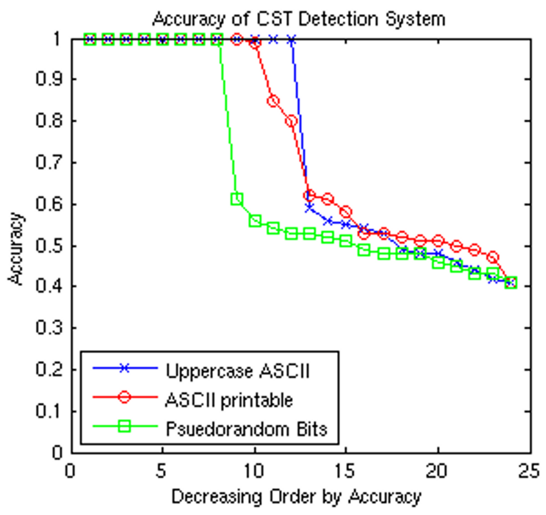
We have presented a stego-system which generates stego-objects using context sensitive tiling. We have analyzed the encoding capacity and the robustness of this system under four example production rule sets and found that variable length encodings offer greater capacity, but that fixed length encodings are far more robust to tampering. To further increase the robustness to tampering, we found that repeating the message can be very effective. The security of the system was experimentally tested against several existing image steganalysis techniques and it was found that these techniques completely



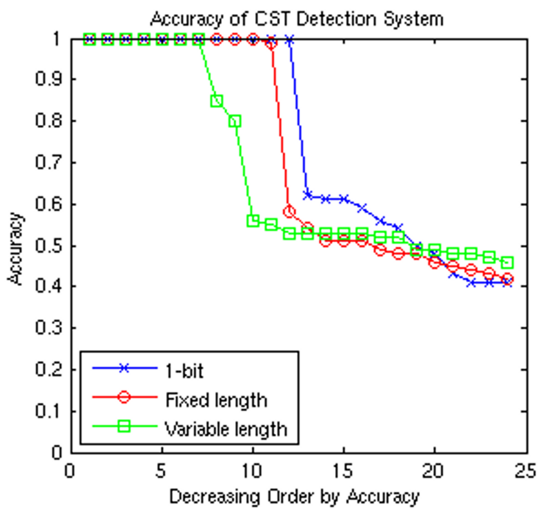
(a) Conditioned on Use of Seed



(b) Conditioned on Rule Set



(c) Conditioned on Embedding Type



(d) Conditioned on Embedding Type

Figure 7. Detection accuracy results for Chi-Square Test.

failed at the task of distinguishing stego objects from cover objects. To attempt to circumvent this problem, two directed steganalysis techniques meant specifically for tiling systems were developed and investigated. The results show that the steganalysis methods are only accurate when no seed is used. In every other case, the directed steganalysis methods performed similarly to random chance.

The work has led us to make some useful conclusions. In general, tampering can cause severe complications at the decoding end, but it is possible to enhance robustness through choice of encoding-type and redundancy. Second, in combination with a shared random number generator for symbol rotation, it can be shown that such synthetic stego is virtually undetectable (a result, motivated by this work, that is proved in a related work by the authors). Finally, the tiling framework can be generalized to many other settings. The system is not limited to creating images and can be used to create text, audio and physical compositions just as easily. It is also possible to generalize the existing stego-system implementation by including support for user-defined tiling systems.

7. **Acknowledgments.** This research is supported by NSF CNS-0716398 and NSF CCF-0939370.

REFERENCES

[1] D. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*, Scribner, USA, 1996.

[2] G. J. Simmons, The history of subliminal channels, *IEEE Jouranl of Selected Areas in Communication*, vol. 16, no. 4, pp. 452-462, 1998.

[3] N. F. Johnson and S. C. Katzenbeisser, A survey of steganographic techniques, *Proc. of Information Hiding Techniques for Steganography and Digital Watermarking*, pp. 43-78, 2000.

[4] C. Blundo and C. Galdi, Hiding information in image mosaics, *The Computer Journal*, vol. 46, no. 2, pp. 202-212, 2003.

[5] P. Sallee, Model-based steganography, *Proc. of the 2nd International Workshop on Digital Watermarking*, pp. 154-167, 2004

[6] R. Radhakrishnan, M. Kharrazi, and N. Memon, Data Masking: A New Approach for Steganography?, *Journal of VLSI Signal Processing*, vol. 41, pp. 293-303, 2005.

[7] A. H. Sung, G. R. Tadiparthi, and S. Mukkamala, Defeating the current steganalysis techniques (robust steganography), *Proc. of the International Conference on Information Technology: Coding and Computing*, vol. 1, pp. 440-444, 2004.

[8] G. R. Tadiparthi and T. Sueyoshi, StegAnim-a novel information hiding technique using animations, *Journal of Engineering Letters*, vol. 13, no. 3, pp. 225-235, 2006.

[9] J. C. Martin, *Introduction to Languages and the Theory of Computation*, McGraw-Hill Higher Education, USA, 1997.

[10] A. Munoz, Stegsecret, 2007, `http://stegsecret.sourceforge.net`.

[11] M. Wegrzyn, Virtual steganographic laboratory, 2009, `http://vsl.sourceforge.net`.

[12] K. Hempstalk, Digitial Invisible Ink Toolkit, 2006, `http://diit.sourceforge.net`.

[13] J. Fridrich, M. Goljan, and R. Du, Detecting LSB steganography in color and gray-scale images, *IEEE Multimedia*, vol. 8, no. 4, pp. 22-28, 2001.

[14] A. Westfeld and A. Pfitzmann, Attacks on steganographic systems Breaking the steganographic utilities ezStego, jsteg, steganos, and s-tools and some lessons learned, *Proc. of the 3rd International Workshop on Digital Watermarking*, pp. 61-67, 2000.

[15] I. Avcibas, M. Kharrazi, N. Memon, and B. Sankur, Image steganalysis with binary similarity measures, *EURASIP Journal on Applied Signal Processing*, pp. 2749-2757, 2005.

[16] S. Katzenbeisser and F. A. Petitcolas, editors, *Information Hiding: Techniques for Steganography and Digital Watermarking*, Artech House, USA, 2000.

[17] P. Vit, On the equivalence of certain markov chains, *Journal of Applied Probability*, vol. 13, no. 2, pp. 357-360, 1976.

[18] A. Kerckhoffs, La cryptographie militaire, *Journal des Sciences Militaires*, vol. 9, pp. 5-38, pp. 161-191, 1883.