A Steganographic Embedding Undetectable by JPEG Compatibility Steganalysis[*]

Richard E. Newman[1], Ira S. Moskowitz[2], LiWu Chang[2]
and Murali M. Brahmadesam[1]

[1] CISE Department
University of Florida
Gainesville, FL 32611-6120
USA
nemo@cise.ufl.edu
&
[2] Center for High Assurance Computer Systems, Code 5540
Naval Research Laboratory
Washington, DC 20375
USA
moskowitz@itd.nrl.navy.mil

Abstract. Steganography and steganalysis of digital images is a cat-and-mouse game. In recent work, Fridrich, Goljan and Du introduced a method that is surprisingly accurate at determining if bitmap images that originated as JPEG files have been altered (and even specifying where and how they were altered), even if only a single bit has been changed. However, steganographic embeddings that encode embedded data in the JPEG coefficients are not detectable by their JPEG compatibility steganalysis. This paper describes a steganographic method that encodes the embedded data in the spatial domain, yet cannot be detected by their steganalysis mechanism. Furthermore, we claim that our method can also be used as a steganographic method on files stored in JPEG format. The method described herein uses a novel, topological approach to embedding. The paper also outlines some extensions to the proposed embedding method.

1 Introduction

Steganography and steganalysis of digital images is a cat-and-mouse game. Ever since Kurak and McHugh's seminal paper on LSB embeddings in images [10], various researchers have published work on either increasing the payload, improving the resistance to detection, or improving the robustness of steganographic methods [1, 15, 21, 22]; or conversely, showing better ways to detect or attack steganography [5, 7, 23]. Fridrich, Goljan and Du recently raised the bar for embeddings in the spatial domain with the introduction of their "JPEG compatibility" steganalysis method, which is very precise at detecting even small

---

changes to bitmap images that originated as JPEGs [6]. This paper presents a steganographic embedding that encodes the embedded data in the spatial domain (bitmap) by manipulating the image in the frequency domain (the JPEG coefficients) (and in fact, the stego image may be stored either as a bitmap or as a JPEG). Due to this, the embedding cannot be detected by JPEG compatibility steganalysis. However, in order to elude other means of detection (either by human inspection or statistical tests), we found it necessary to introduce notions of topology. This, we believe, has larger implications.

Often, one is called upon to perform steganalysis on the uncompressed, spatial realization (from, e.g., a TIFF, BMP, or PNG file) of an image (i.e., its bitmap). Eggers, Bäuml and Girod [4] assert that "...uncompressed image data looks to Eve as suspicious as encrypted data. Thus, the steganographic image r has to be always in a compressed format." However, many small images are stored in bitmap format without compression, and larger images may be stored with lossless compression. Also, steganography can be used to store data on a local disk without placing it on a website or sending it through email, in which case there are many instances in which uncompressed formats may be found. A user may choose to store passwords or other secret information in her local image files in a way that she can recover them but another might not even know that they were there at all. In many cases, naïve users will transfer data in arbitrary formats. Even sophisticated users may find that their recipients do not have software compatible with the format of choice, and so either send an alternative format or provide a choice of formats (much as websites often provide both postscript and PDF versions along with a compressed version of a document). Further, Eggers et al. earlier posit that Eve should consider any natural image data as suspicious, given its ability to hide a considerable amount of embedded data. However, they concede that this at the very least makes the field less interesting, and we contend that transmission of all sorts of images in every conceivable format over the Internet is likely to continue, with the mix mostly consisting of innocent images. Therefore, it is still incumbent upon the steganalyst to detect which images are innocent and which are suspicious on the basis of something other than their format alone; that is, we discount "cover format profiling."

A common steganographic method for modifying such images is replacing the least significant bits (LSBs) with the embedded information [8, 10, 12]. Since the image is stored in a lossless format, there is much redundancy of which steganographic methods can take advantage. Although steganographic methods that replace lower bit planes in the spatial domain are easily detectable by statistical tests, steganographic methods that affect the lower bit values of only a small percentage of the pixels (e.g., [17]) are extremely difficult to detect (e.g., [16]) by statistical means. However, if the image was at one time stored as a JPEG, the artifacts of the quantization of the DCT coefficients remain in the spatially realized bitmap. Leveraging this fact, JPEG compatibility steganalysis may detect even such minuscule tampering of the bitmap derived from a JPEG.

With this in mind we feel that it is important to come up with a way of "tricking" such steganalysis tools, and thus allow a modest amount of information (payload, e.g. [15]) to be embedded in the spatial realization of a JPEG, without detection. This paper presents one such method.

Fridrich, Goljan and Du never stated or implied that their method could not be evaded. The beauty of their steganalysis method is that it shockingly showed how small deviations in the bitmap of a JPEG could be detected. We show in the body of this paper that our method will not be detected by the JPEG compatibility method of Fridrich et al. We also believe that for small payloads our method with simple extensions will not be detectable by any other existing steganalytical tools. *A priori* our steganography is performed in the spatial domain, that is, the data are embedded by encoding in the spatial domain, but since the changes actually come from adjusting quantized DCT coefficients our method *a fortiori* can actually use JPEG files (i.e., an image saved in the JFIF format as a .jpg) as the cover/stego file. Our method is a hybrid in that it encodes the embedded data in the spatial domain via JPEG coefficient manipulation; this is why it is resistant to detection using either spatial or frequency steganalysis techniques.

Marvel et al. [14] propose a simple means of storing one bit per block in the quantized JPEG coefficients. Although this bears some surface resemblance to the work given here, the embedded data are stored in the JPEG coefficients themselves, and it is required that the receiver have them in order to extract the embedded data. In the method presented here, the data are encoded in the spatial domain, albeit through manipulation of the frequency domain, and the receiver must have the spatial domain realization of the image in order to extract the embedded data. In our baseline system, the receiver must also generate the topologically nearby spatial blocks in order to determine whether the block in question actually encodes data or is unusable. In follow-on work, this requirement is eliminated.

Another contribution of this paper, and perhaps an even more important one, is that we introduce a topological approach to steganography. We attempt to formalize what it means for one image (or part of an image) to be near another image (or part of an image). The obvious upshot of this is that the stego image will be indistinguishable from the cover image (by either human or machine).

Section 4 presents a baseline, proof-of-concept version of our method of hiding in the bitmap (spatial) realization of a JPEG file that is not detectable by the method of Fridrich et al. [6]. Our method makes use of the topological concept of "closeness," which is formalized along with a generalized form of our method in section 5. Extensions are discussed in section 7, with results in section 6 and concluding remarks in section 8. Section 3 presents and analyzes the method of Fridrich et al. and see that it can be "tricked," provided you retain qualities that a legitimate JPEG should have in the spatial domain. In order to appreciate the method of JPEG compatibility steganalysis of Fridrich et al., and to understand our way around it, it is necessary to have a basic understanding of how JPEG works. Section 2 presents a brief discussion of JPEG for completeness.

2   JPEG Basics

JPEG [19] first partitions a bitmapped image (such as one might obtain from a CCD camera or a scanner) into 8 by 8 blocks of pixels, starting with the top, leftmost pixel. Generally, the pixel values are constrained to one or a few planes of one or a few bits (e.g., 8- or 10-bit grayscale, or 24-bit color). Each of these 64-pixel blocks (A in Figure 1) in the spatial domain is then transformed using the Discrete Cosine Transform (DCT) [24] into the frequency domain, which produces 64 raw DCT coefficients per plane (B). The resulting coefficients are real numbers (albeit over a limited range), and so require considerably more storage. Each coefficient is then divided by the quantum step for that coefficient (defined by a quantizing table $QT = QT[1], ..., QT[64]$) and rounded to the nearest integer to produce quantized coefficients (C) (JPEG coefficients). This step (including rounding) is generally called quantization. Lossless entropy coding further reduces the space needed to store the quantized coefficients considerably, and is the bulk of what is stored in .jpg files (X). The "quality level" of JPEG compression determines the magnitude of the quantum steps in the quantizing table, which in turn determines the visual quality of the compressed image after decoding. To decode a JPEG file, the inverse DCT (IDCT) is applied to the decompressed, dequantized coefficients (i.e., the JPEG coefficients, C, are multiplied by their respective quanta to produce the integer multiples of the quanta nearest to the original coefficients, D) to obtain a raw bitmapped output file (E) in the spatial domain, whose pixel values are real numbers. These values are then clamped (if they are less than the minimum value or greater than the maximum value for the format used) and rounded to the nearest integer value in the range $[0..2^n - 1]$ to produce the final output block (F).

Although the Discrete Cosine Transform is mathematically invertible (i.e., for a block A, $IDCT(DCT(A) = A)$), quantizing and dequantizing by any value other than unity generally distorts the DCT coefficients so greatly that, referring to Figure 1, A may not be the same as F. Likewise, clamping and rounding render the the process of decoding and then re-encoding imprecise, even when the same quantizing table is used. That is, referring to Figure 1, C and C' (and hence, D and D') may not be identical, even if $F' = F$.

It is important to note that at any non-trivial quantization level, there are many bitmap blocks that cannot be the output of JPEG decoding at all. We will call the spatial domain blocks that *are* the result of decoding a set of JPEG coefficients for a given quantization table JPEG compatible blocks, or JPEG blocks for short, and those that are not, JPEG incompatible, or non-JPEG blocks.[3]

---

[3] Note that blocks that are JPEG incompatible for one quantizing table may be JPEG compatible for a different quantizing table, and vice-versa.
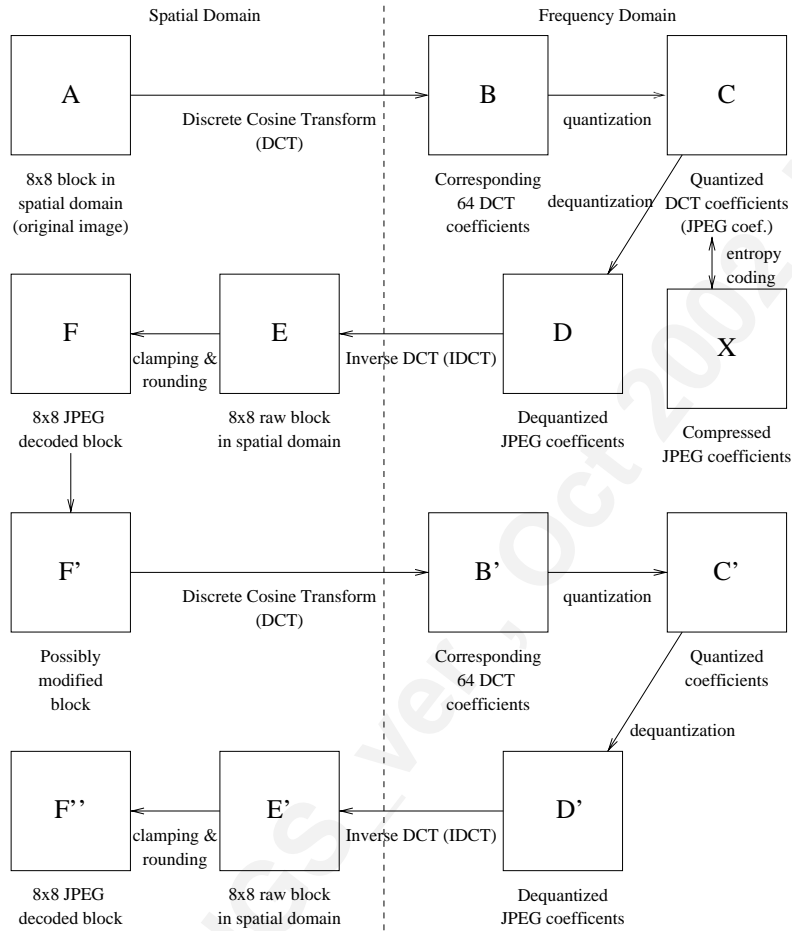
Fig. 1. JPEG Operation.

3 JPEG Compatibility Steganalysis

Fridrich, Goljan and Du [6] introduced an ingenious steganalysis technique that determines whether a bitmap representation of an image derived from a JPEG file has been altered. If a bitmap image were derived from an image once stored in JPEG format, their method can determine this in most cases, even if the low order bits of the image have been manipulated after conversion to bitmap format. Their method takes advantage of the last fact mentioned in the previous section: not all spatial domain blocks can be the output of decoded JPEG coefficient sets, i.e., not all spatial blocks are JPEG blocks. Their steganalytical method first determines that the bitmap was at one time stored in JPEG format, then recovers the 8×8 JPEG block alignments and the best candidate for the quantization table. It then detects those blocks that could *not* have been produced by the

JPEG decoding process. Since changing a single bit in a spatial block can cause a JPEG block to become JPEG incompatible, this approach is extremely sensitive to manipulation of images in the spatial domain; it can readily detect even low payload size [15] steganographical embeddings that do not take the JPEG characteristics into account, provided they manipulate bitmaps that were once stored in JPEG form.
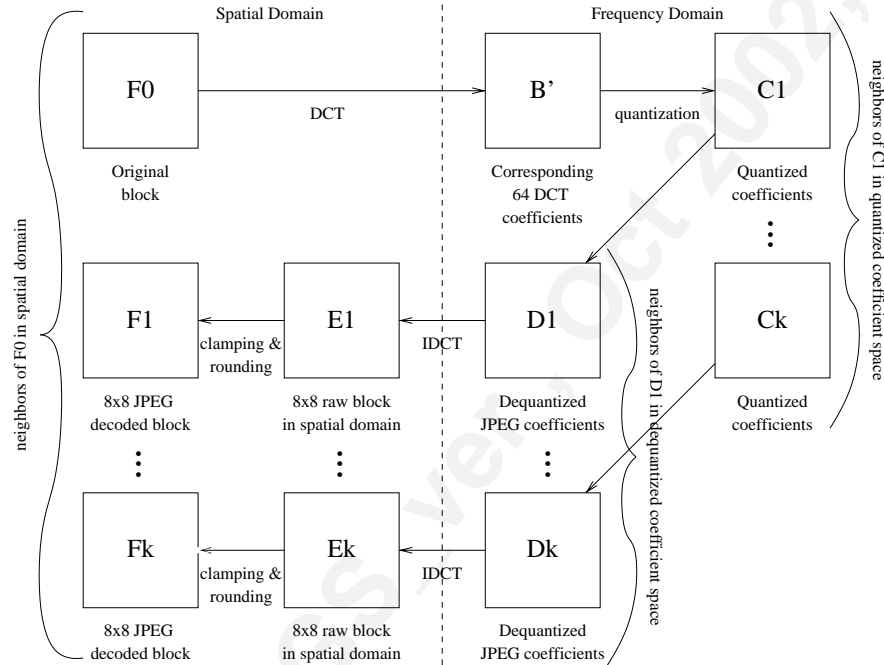

Fig. 2. JPEG-Compatible Neighbors of a Spatial Block.

As they note, their method has some limitations, the most notable being the cases of blocks in which clamping has occurred (i.e., the JPEG decoding intermediate block E held values less than -0.5 or greater than 255.5, for which the rounding error is greater than 0.5), and when the JPEG quality is very high (i.e., when the quantization table has very small values). In the former case, the basic test they use for energy bounds does not apply, while in the latter, the number of possible sets of DCT coefficients they must test is prohibitively large (although theoretically possible). In addition to these cases, there are some other cases in which an image has been manipulated, but the manipulation will not be detected by their method even though artifacts may be apparent to a human observer.[4] Nonetheless, any bitmapped file that was once stored in JPEG format that fails their test can only do so if it has been manipulated, and their test is

---

[4] An example is given at the beginning of the next section.

sensitive enough to detect even a single bit change in a bitmap file. Hence their test produces no false positives, but can produce some false negatives.

Any steganographic embedding that embeds data directly in the JPEG coefficients will not be detected by JPEG compatibility steganalysis, as the decoded spatial image will consist entirely of JPEG blocks. However, these embedding methods use .jpg files as the stego image storage format, so that the JPEG coefficients are maintained without error after the embedding is performed. If a bitmap image is produced for the stego image, then it must be re-encoded in JPEG form in order to recover the JPEG coefficients, which is a process likely to introduce errors in the steganographic data extraction process. In the next section, we present a steganographic spatial embedding method that is JPEG compatible. In fact, the stego image may be stored in either bitmap format or as a JPEG.

4   A Baseline Spatial Domain Stego embedding that Defies JPEG Compatibility Steganalysis

This section presents our baseline version of a novel, topological approach that may change many bits in the spatial file, but will never be detected by JPEG compatibility steganalysis; it will always produce a false negative. Extensions will be explored in section 7. The basic idea is to manipulate the image in such a way that all of the 8×8 blocks are valid outputs of the JPEG decoder, and all the spatial blocks are "near" the original spatial blocks as well. Of course, a file that is the result of intermingling 8×8 blocks from two different decoded JPEG files that both used the same quantization table would satisfy the first condition, but that is likely to be easily detected by the human visual system (HVS). The key is to be able to escape detection by either the HVS or machine, which means making the result compatible both with JPEG and with the HVS.

This section will introduce the topological concept of neighbor, and will define "rich" and "poor" blocks according to the ability of the system to use them to embed data. In the context of this discussion, neighbors of a block will not be the surrounding blocks in the image file (compressed or not), but will be other blocks that are not much different in their content from the block of interest (that is, they are intended to be undetectably different to the steganalyst). Those blocks that are in effect indistinguishable from the block of interest will be called its neighbors, and a block will be called rich if it and its neighbors can encode any datum desired; otherwise it will be called poor.

Our baseline system stores only one bit of embedded data per JPEG block, in 8-bit, grayscale images. It uses the LSB of the upper left pixel in the spatial block to store the embedded data. A small, fixed size length field is used to delimit the embedded data. As a first cut, if the bit is the desired value, then we could leave the block alone. If the desired bit is the opposite of the original value, then the system changes the JPEG block in such a way that the upper left LSB is the desired value, *but the modified spatial block is still JPEG compatible.* However, as we will see, there is more to it than this.

Encoding is done by going back to the quantized coefficients for that JPEG block and changing them slightly in a systematic way to search for a minimally perturbed JPEG compatible block that embeds the desired bit (one of the F_j's in Figure 2), hence the topological concept of "nearby." This is depicted in Figure 2, where B' is the raw DCT coefficient set for some block F_0 of a cover image, and D_1 is the set of dequantized coefficients nearest to B'.[5] Note that B' is a point in (continuous) raw DCT coefficient space, while each D_i is a point in the subspace consisting of dequantized JPEG coefficients (for the quantization table in use). In other words, the neighbors F_i, $i = 1, 2, ..., k$ of the spatial block F_0 must be blocks that are the decoded JPEG output of points near D_1 in the dequantized coefficient block space. Thus, our maps are "continuous" (in the topological sense).

The topological concept of nearness has to be defined in terms of both human detection and machine detection, and this is the subject of continuing research. The preliminary version presented here changes only one JPEG coefficient at a time by only one quantization step. In other words, it uses the L_1 metric on the points in the 64-dimensional quantized coefficient space corresponding to the spatial blocks, and a maximum distance of unity. (Note that this is different from inverting the LSB of the JPEG coefficients, which only gives one neighbor per coefficient.) For most blocks, a change of one quantum for only one coefficient produces acceptable distortion for the HVS. This results in between 65 and 129 JPEG compatible neighbors[6] for each block in the original image.

If there is no neighboring set of JPEG coefficients whose spatial domain image carries the desired datum, then the system could deal with this in a number of ways. One is to treat this as an error in the stego channel and provide error correction to handle it. Another is to provide some kind of signal that this block is not to be used (that is, in the embedded data stream, insert a control sequence in the bits preceding the unusable block(s) to indicate that it is (they are) not to be used, then move the data that would have been encoded there to usable blocks occurring after the unusable block or blocks). Yet another is to provide a map of the locations of the unused blocks within the embedded data. A similar approach is used in BPCS steganography [8] to identify blocks for which the embedded data were transformed so they would be correctly identified as embedded data by the receiver. These approaches use up scarce payload space of the steganographic encoding. A fourth approach trades off computation at both sender and especially the receiver for improved payload space. The sender

---

[5] For quantized DCT coefficients or for DCT coefficient sets, dequantized or raw, we will use the L_1 metric to define distances. D_1 is the set of coefficients of B' rounded to the nearest multiple of the corresponding quantum in QT. The notion of neighbor will be made precise in subsection 5.1.

[6] Each of the 64 JPEG coefficients may be changed by +1 or -1, except those that are already extremal. Extremal coefficients will only produce one neighbor, so including the original block itself, the total number of neighbors is at most 129, and is reduced from 129 by the number of extremal coefficients. If the QT has very small values, it is possible that some of the neighbors coincide, reducing this number further, but for typical quantum values, this is unlikely.

avoids unusable blocks in such a way that the receiver can tell which blocks the sender could not use without the sender explicitly marking them. This is the method our baseline system employs.

There are two criteria that must be met for this approach to work. First, the receiver must be able to test each block that it receives to determine whether it has been used to encode data or not. Second, if the receiver classifies a block as having been used to encode data, it must encode the correct datum. If the set of neighbors that the sender explores to find a suitable block does not have some block that could send any possible desired datum, then that block might be considered to be inutile. Thus the sender and the receiver could agree that a block can be used to encode data if and only if, for any possible datum, its set of neighbors includes at least one block that can send that datum. We will call these blocks 'rich,' and those that do not satisfy this criterion 'poor.'

However, for the receiver this decision is based on the block received, to wit, the block with which the sender replaced the original block. This in turn means that the sender can not just find any neighboring block that encodes the desired datum (or leave a block alone if it already conveys the desired datum), but must also test a candidate replacement block to see if the receiver would consider it to have been used (i.e., would find it to be rich). That is, the neighborhoods are not a partition, and rich blocks are not guaranteed to have only rich neighbors. Otherwise, if the sender chooses the original or a neighboring block to encode a datum, but that chosen block is not rich (i.e., there is some datum that its set of neighbors can not encode), then the receiver will mistakenly assume that the replacement block was not used and will skip it.

As long as there is at least one rich block that conveys the desired datum among the neighbors of the original block, then that block can be used to replace the original (even if it conveys the same data, so that an original but poor block that conveys the correct data can be replaced by one of its rich neighbors that conveys the same data, if one exists). Otherwise, the block cannot be used by the sender. However, if the original block is rich, and hence appears to be usable, then it must not be left untouched or else the receiver will classify it as used and include the datum it encodes in the received stream in error. Instead, the original block must be replaced by one of its poor neighbors that will be classified as unused by the receiver, regardless of what datum it may encode. These notions are formalized in the next section.

5 Generalization and Formalization of Our Stego Embedding Technique

This section describes formally how our method hides an arbitrary embedded data string in the spatial realization of a JPEG image. The embedded data must be self-delimiting in order for the receiver to know where it ends, so at least this amount of preprocessing must be done prior to the embedding described. In addition, the embedded data may first be encrypted, and it may have a frame check sequence (FCS) added if unusable blocks are rare to save the receiver from

costly tests, allowing it to assume that all the blocks were used unless the FCS fails.

Let the embedded data string (after encryption, end delimitation, frame check sequence if desired, etc.) be $s = s_1, s_2, ..., s_K$. The data are all from a finite domain $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_N\}$, and $s_i \in \Sigma$ for $i = 1, 2, ..., K$. Let $\tau : \Sigma^* \to \{0, 1\}$ be a termination detector for the embedded string, so that $\tau(s_1, s_2, ..., s_j) = 0$ for all $j = 1, 2, ..., K - 1$, and $\tau(s_1, s_2, ..., s_K) = 1$. Let $S = [0..2^m - 1]^{64}$ be the set of 8×8 spatial domain blocks with m bits per pixel (whether they are JPEG compatible or not), and let $S_{QT} \subseteq S$ be the JPEG compatible spatial blocks for a given quantization table QT. Let Φ extract the embedded data from a spatial block F,

$$\Phi : S \to \Sigma$$

We polymorphically extend this to sets of blocks $\Gamma \subseteq 2^S$ by

$$\Phi(\Gamma) \stackrel{\text{def}}{=} \{\Phi(\gamma) \mid \gamma \in \Gamma\}.$$

Let μ be a pseudo-metric[7] on S_{QT},

$$\mu : S_{QT} \times S_{QT} \to \mathbb{R}^+ \cup \{0\}.$$

Let $N_\Theta(F)$ be the set of JPEG compatible neighbors of JPEG compatible block F according to the pseudo-metric μ and threshold Θ based on some acceptable distortion level (μ and Θ are known to both sender and receiver),

$$N_\Theta(F) \stackrel{\text{def}}{=} \{F' \in S_{QT} \mid \mu(F, F') < \Theta\},$$

where QT is the quantizing table for the image of which F is one block. The $N_\Theta(F)$ may be thought of as a basis for the "topology," however our technique only uses a fixed Θ which is chosen small enough so that the HVS cannot detect our stego embedding technique. Neighborhoods can likewise be defined for JPEG coefficients and for dequantized coefficients for a particular quantizing table (by pushing the pseudo-metric forward).

If $F' \in N_\Theta(F)$, we say that F' is a *neighbor* of F (the Θ is understood and not explicitly mentioned for notational convenience). Being a neighbor is both reflexive and symmetric. Now we can make our definitions from the previous section precise.

5.1 Definitions

Definition: A block F is called *rich* if and only if

$$\Phi(N_\Theta(F)) = \Sigma,$$

---

[7] That is $F = F' \Rightarrow \mu(F, F') = 0$ but not necessarily the converse. See Chap. 9, Sec. 10 of [3]. This is needed because nonlinearities introduced by rounding in both the quantization step and in decoding can possibly cause two distinct, JPEG compatible, spatial blocks to have distance 0. For most JPEG quantizing tables, however, μ is in fact a true metric.

that is, for every datum $\sigma \in \Sigma$, F has at least one neighbor, F', that encodes σ, and we write $F \in R$ (the set of rich blocks). Otherwise, F is *poor*.

Definition: Block F is *usable* if and only if for every datum $\sigma \in \Sigma$, F has at least one neighbor that both encodes σ and is rich:

$$\Phi(N_\Theta(F) \cap R) \ = \ \Sigma \ .$$

If F is not usable, then it is *unusable*. (In Section 7, we relax this definition somewhat.) Of course any usable block is rich, but the converse need not hold.

Claim 0: If block F is unusable then either F is poor, or one its neighbors is poor.

Proof of Claim 0: F is either rich or poor. If F is poor we are done. Assume then that F is rich, therefore one can always find a neighbor of F that encodes σ for any $\sigma \in \Sigma$. If every such neighbor were rich, then F would be usable, which it is not. Therefore, when F is rich, there exists some neighbor of F that is poor.

5.2 Algorithm in brief

The key to our method is that the *receiver only considers rich blocks for decoding*. The receiver ignores poor blocks — it simply skips over them. If the transmitter has a poor block it is sent and the receiver ignores it. Thus, no information is passed if a poor block is transmitted.

- **transmitter has usable block** (F is usable):
  - If F encodes the information that the transmitter wishes to send, the transmitter leaves F alone and F is sent. The receiver gets (rich) F, decodes it and gets the correct information.
  - If F does not encode the correct information, the transmitter replaces it with a rich neighbor F' that does encode the correct information. The replacement ability follows from the definition of usable. Since F' is a neighbor of F the deviation is small and the HVS does not detect the switch.
- **transmitter has unusable block** (F is unusable):
  - If F is poor, the transmitter leaves F alone, F is sent, and the receiver ignores F. No information is transferred.
  - If F is rich, the transmitter changes it to a neighbor F' that is poor. The ability to do this follows from Claim 0. Block F' is substituted for block F, the receiver ignores F' since it is poor, and no information is passed. Since F' is a neighbor of F the deviation is small and the HVS does not detect the switch.

Note that when dealing with an unusable block that the algorithm may waste payload. For example, if F is unusable and poor, F may still have a rich neighbor that encodes the desired information. See section 7 for further discussion. The advantage of the algorithm as given above is that it is non-adaptive. By this we mean that the payload size is independent of the data that we wish to send. If we modify the algorithm as suggested, the payload can vary depending on the data that we are sending.

5.3 Algorithm in detail

To hide the embedded data, the sender first must find a JPEG image cover file I with at least K usable blocks. (Since the sender has great flexibility here, it should not be difficult to find such an image if the total number of blocks M is sufficiently larger than K, Σ is not too large, and Θ is not too small.) Let the spatial domain JPEG blocks of the cover file be $I_1, I_2, ... I_M$, and let π be a permutation of the block indices known to the sender and receiver so that the blocks of I are considered in the permuted order, $I_{\pi(1)}, I_{\pi(2)}, ... I_{\pi(M)}$. (Note that the order in which blocks of I are tested for use must be known to both sender and receiver, so that the receiver extracts only the blocks that were used and extracts them in order. This permutation can be part of the key material or derived from it. Our baseline system scans blocks in left-to-right, top-to-bottom order.) Let the usable blocks of the permuted order of I be $V = V_1, V_2, ..., V_{M_1}$, and let the unusable blocks of the permuted order of I that are interspersed with V be $U = U_1, U_2, ..., U_{M_2}$. Thus $M = M_1 + M_2$, $M_1 \geq K$, and either $I_{\pi(1)} = V_1$ or $I_{\pi(1)} = U_1$.

For the ith datum, s_i, $i = 1, 2, ..., K$, the sender will pick some rich block $V_i' \in N_\Theta(V_i)$ such that $\Phi(V_i') = s_i$. The sender will then replace each usable block V_i with block V_i' in forming the stego image I' (note that if $\Phi(V_i) = s_i$ and $V_i \in R$, then $V_i' = V_i$, i.e., the block need not be replaced since the receiver will correctly decode it already).

For each unusable block U_i of I that is interspersed with the blocks used to embed the embedded data, the sender will either leave U_i alone in forming I' if U_i is poor, or will replace U_i with a poor neighbor U_i' otherwise. Claim 0 tells us we can do this.

The receiver then tests the blocks of the stego image I' in the predefined order $\pi(1), \pi(2), ...$, discarding the poor blocks $U_1', U_2', ...$ and extracting the rich blocks of I' (note that they do not have to be usable), $V_1', V_2', ..., V_K'$ to extract the embedded data, $s_i' = \Phi(V_i')$. This continues until the last datum, s_K', is extracted, and s' is found to be complete by the self-delimiting mechanism, $\tau(s') = 1$. The remainder of I' is ignored.

5.4 Claims

Claim 1: JPEG compatibility steganalysis will not detect this stego embedding method.

Proof of Claim 1: Since every block in I' is a valid JPEG block (of course with the same quantization table), the JPEG based steganalysis can not detect that it has been altered.

Note that if the pseudo-metric μ/threshold Θ are not defined/chosen properly, there may be other means (even human inspection of the image) that could detect artifacts indicating that I' is a stego image.

Claim 2: Any usable block F has a neighbor that can encode any datum σ^* in such a way that the receiver will accept it.

Proof of Claim 2: F is usable $\iff$ $\forall \, \sigma \in \, \Sigma, \, \exists F' \in N_\Theta(F) \cap R, \sigma = \Phi(F')$ by definition. In particular, $\exists \, F^* \, \in N_\Theta(F) \cap R$ such that $\sigma^* = \Phi(F^*)$. Since $F^* \, \in \, R$, the receiver will classify the corresponding block I_i of I as rich, and will extract the datum σ^* from it.

Claim 3: Any unusable block F will be modified (if necessary) in forming the stego image I' so that the receiver rejects it.

Proof of Claim 3: By Claim 0, either F or one of its neighbors is poor. If F is poor, leave it alone and the receiver rejects it for decoding. If F is rich we replace it with one of its poor neighbors, which the receiver then rejects.

Claim 4: Using the stego embedding described above, a cover file I with at least K usable blocks can embed any self-delimited data string $s = s_1, s_2, ..., s_K$ correctly.

Proof of Claim 4: While space limitations preclude us from presenting the full proof here, it is easily shown by induction on the length of the embedded string and is in the appendix.

6   Results

We have implemented the baseline version of our method. As discussed in section 4, this initial version is very rudimentary and is essentially a proof of concept. It does, however, yield very good results that are resistant to detection. Since the changes to the JPEG coefficients are minimal (at most one quantum of one coefficient), and the quanta have been chosen to more or less equalize the effect on the HVS, the stego image is indistinguishable from the cover by humans. Changes to the statistics of the JPEG coefficients are minimal by design. An initial version had a bias toward incrementing the coefficients, which caused the number of JPEG coefficients with a value of 1 to outnumber significantly those with a value of -1. Since this asymmetry could have been detected easily, we removed this bias in our baseline system. In both versions, the JPEG coefficient frequencies decreased away from zero, and were generally concave upward, and so would pass the test of Westfeld et al. [25]. Further, although there are typically a large number of zero coefficients that are changed (since these predominate), the relative number is small (usually around half of a percent). Thus, we expect that statistical tests (such as correlation toward one) will fail to discern an abnormality. An example of the baseline embedding for a particular block is given in Figure 3. (We see little point in taking up space showing two spatial images that look the same when printed at poor resolution.) A specific JPEG coefficient block results in the spatial block (cover image) on the left of Figure 3. We desire that the LSB of the upper left pixel be 0 (which it is not). Therefore we adjust the JPEG coefficient block by one quantum (we change the sixth JPEG coefficient $AC_{0,2}$ from 0 to -1), which results in the spatial block (stego image) on the right (in which the LSB of the upper left pixel is 0).

```
Original Spatial Block                    Spatial Block after Embedding
**********************                    *****************************
137 137 137 135 132 127 123 121          136 137 137 136 133 128 123 119
136 136 135 134 131 128 124 122          135 135 136 135 133 128 124 121
134 134 133 133 131 128 126 125          132 133 134 134 132 129 126 123
132 132 131 131 130 129 127 127          131 131 132 132 131 129 127 126
131 131 130 129 128 128 128 128          130 130 130 130 130 129 127 127
132 131 129 128 127 127 127 128          131 130 130 129 128 127 127 127
133 132 129 127 126 126 126 127          132 131 130 128 127 126 126 126
134 132 129 127 125 125 125 126          133 132 130 128 126 125 125 125
```

Fig. 3. Cover image spatial block and stego image spatial block

Our baseline version runs somewhat slowly due to the number of tests[8] that are made and the computational burden of each test. With typical JPEG files, however, and encoding only one bit per usable block, the number of tests it has to make is small since it only has to find neighboring blocks that encode both values, and with typical quanta, these are quickly found. The payload is small — only one bit per usable 8×8 block, but the likelihood of detection is very low. Although this already small number may be decreased by the number of poor blocks found in the cover image, with typical JPEG files we find very few poor blocks, so this is not an issue.

7   Extensions

Although the current definition of usable does not depend on the datum that the block is intended to encode (and thus is independent of the embedded data s), there may be greater payload space available if the definition is loosened to be specific to a particular datum σ (refer to the discussion at the end of subsection 5.2).
Definition: A block F is *usable for datum* σ if and only if F has at least one neighbor that both encodes σ and is rich:

$$\{\sigma\} \cap \Phi(N_\Theta(F) \cap R) \neq \emptyset.$$

This allows a block that is not usable itself to be usable for σ if it has a rich neighbor, possibly itself, that encodes the desired datum. However, it should be noted that this makes the embedding adaptive to both the image *and* the embedded data, so that the payload size becomes dependent on the embedded data (as well as the cover image—same as before). The degree to which this increases the payload space by decreasing the probability of encountering an unusable block is worthy of exploration.

If unusable blocks are rarely encountered, then it may be desirable to have an error detection code appended to the whole message so that the receiver can determine if there were any unusable blocks or not, and search for them only if

---

[8] It also does much computation to gather statistics that would not be needed simply to perform embedding or extraction.

there were. This is relatively inexpensive in terms of space (a short CRC will do), and only if the test fails must the receiver perform the more expensive block-by-block test for usability. The additional work required of the receiver to check the CRC is minimal, and all of the decoding work it performs would have to be done anyway, so this extension is likely to provide a significant gain in decoding speed at very little cost.

Our baseline system only works on grayscale images; it is easily extended to color (multichannel) images. While currently the stego image is stored, sent, or posted in bitmap format (e.g., TIFF, BMP), we have enhanced the system with an option to store the stego image in JPEG format as is done in spread spectrum steganographic techniques [13, 20] and other JPEG-based systems such as Jsteg or F5 [25]. This is because our modifications are performed on the quantized coefficient blocks, and then we choose from among the corresponding spatial blocks. It does not suffice simply to reencode the bitmap stego image, as the reencoding may not produce decoded output identical to the stego image. Instead, it is necessary to remember the JPEG coefficients for the replacement blocks, and store these in the format required.

Provos has described methods for detecting information hidden in the JPEG coefficients [21, 22]. In these works, the statistical characteristics of the JPEG coefficients are analyzed to determine if there has been tampering. Based on the results reported, it is unlikely that the small changes our baseline method makes to the JPEG coefficients will be detected (at most one JPEG coefficient per block is changed). Even so, the flexibility afforded by often having more than one choice for the coefficient set with which to encode a datum should allow selection based on minimum disturbance of the coefficient statistics. This will require further investigation.

Currently, the search order and the data extraction function Φ are fixed. Use of a key may provide a means to make this system satisfy Kerckhoffs' principle [2], so that even with knowledge that the system is being used on a subset of images, without the key, detection of which of the images are stego images and which are not is practically impossible. One set of issues as yet to be resolved includes the best way to use the key to define the search order π of blocks in I (and I') and the best way to use the key to define Φ (which may be parametrized to be Φ_i). The key may also contain information used to set μ, Θ and Σ.

However, the main question that remains is how better to construct the pseudo-metric μ and how to pick the threshold Θ that are used to define the neighborhood N_Θ. Our baseline system uses only those JPEG blocks that are the result of decoding the vectors of quantized DCT coefficients that differ from the quantized DCT coefficient set of the original block I_j in only one place i, and thereby only unity. That is, we use the L_1 metric in the JPEG coefficient space with $\Theta = 1 + \epsilon$. This usually provides 129 neighbors for each block (including the block itself), but depending on the number of extremal coefficients, the total may range between between 65 and 129 candidates to replace the block. In most cases, this should be sufficient to encode more than one bit per usable block reliably. We expect that for most blocks and coefficients, we will be able

to change single coefficients by more than one quantum, and will be able to change more than one coefficient simultaneously without introducing humanly detectable artifacts, resulting in a combinatorial number of acceptable neighbors of each original block. A larger neighborhood will allow the approach to encode a larger amount of data per block than a single bit. While a larger Σ allows more bits to be stored per usable block, at the same time it reduces the probability that a block is usable for a fixed Θ. Generally, it is of interest to determine what is the best balance between the size of the data set Σ, the pseudo-metric μ, and the threshold Θ, so that the payload space can be maximized without detection. The pseudo-metric and threshold must be set at least so that the artifacts produced by the replacement of the blocks in the stego image are not obvious to the trained human eye.

The baseline pseudo-metric makes no distinction among the DCT coefficients. However, there are two good reasons it might do so. First, the HVS has different sensitivities to the different coefficients (that is, one can generally change the higher frequency components by greater values than the lower frequency components without human detection). The quantizing tables take this into account by using larger quanta for coefficients to which the HVS is less sensitive, and so the baseline pseudo-metric just relies upon this fact to equalize the changes relative to the HVS. It may be better for the pseudo-metric to consider this more directly. Second, with reasonable compression, many of the quantized DCT coefficients are zero, which is where much of the compression gain is made during entropy coding. If these coefficients are modified, it may be easier for machine detection to discover tampering inconsistent with typical JPEG images (even though the image is entirely JPEG compatible and the overall statistics still appear normal). For these reasons, it may be desirable to restrict the ways in which the JPEG coefficients are changed in a more sophisticated manner.

Beyond this, adaptive encodings should be considered [8, 11, 18]. It would be of interest to explore the degree to which the threshold (and perhaps even the pseudo-metric) may be adapted to each block I_i, so that blocks that contain sufficient amounts of clutter can encode more embedded data, while blocks whose alteration would be more easily detected may encode less data or even no data. The complexity measure used by Kawaguchi et al. may be of use for this [9]. Here, the nature of the block I_i being considered for use affects the threshold Θ_i and possibly the pseudo-metric μ_i. Care must be taken, since these must also apply to the replacement block, which is all that the receiver sees, and from which the receiver must be able to determine μ_i and Θ_i.

It would also be useful to extend this approach to one that is robust in the presence of noise and other alterations to the stego image. One interesting twist is that JPEG-compatibility steganalysis can be used as error correction for some noise introduced in the spatial domain. Using this approach, the original (JPEG-compatible) spatial image can be restored, so that an error-free version of the stego image can be extracted.

8 Conclusions

This paper has briefly discussed JPEG encoding, and the method used by Fridrich et al. to detect tampering with JPEG based bitmap images. It then described a stego embedding method to circumvent detection by the JPEG compatible steganalysis method, including proofs of correctness for the embedding method. While our baseline method is both low rate (1 bit per block) and is easily detectable if the approach is known, it is only a proof of concept. More advanced versions improve the data rate, efficiency, and decrease the detectability of the system (perhaps to the point of satisfying Kerckhoffs' principle).

One might want to use an improved version of this method to store relatively small amounts of data in a relatively undetectable way, or if it is desired to store them in spatial form. Since only one (or a few) coefficients are changed per block, the overall statistical changes will be small, as will be the visual distortion (relative to the distortions already present in the compressed cover, assuming that the QT is balanced in the effect of one quantum change on human perceptibility). The steganalyst is not likely to detect changes in either the frequency domain or the spatial domain, even using extremely sensitive detection methods.

Also, our method can be extended as a steganographic method for files stored in the JPEG format, and detectability in the frequency domain is considered in follow-on work. Equally as important, topological notions of pseudo-metrics and neighborhoods are used to define its operation and as a perspective on the problem. Finally, some extensions to the work are proposed to increase its payload space or decrease the likelihood that an image is correctly detected by steganalysis.

9 Acknowledgments

References

1. R. Anderson. Stretching the limits of steganography. In R. Anderson, editor, *Information Hiding 1996*, volume LNCS 1174, pages 39–48. Springer, 1996.
2. R. Anderson. *Security Engineering*. Wiley, 2001.
3. J. Dugundji. *Topology*. Allyn and Bacon, 1976.
4. J. J. Eggers, R. Bäuml, and B. Girod. A communications approach to image steganography. In *SPIE Electronic Imaging 2002, Security and Watermarking of Multimedia Contents IV*, volume 4675, pages 26–37, San Jose, USA, Jan. 2002.
5. J. Fridrich. Methods for detecting changes in digital images. In *6th IEEE International Workshop on Intelligent Signal Processing and Communication Systems (ISPACS'98)*, Melbourne, Australia, 4-6 November 1998.
6. J. Fridrich, M. Goljan, and R. Du. Steganalysis based on JPEG compatibility. In A. Tescher, B. Vasudev, and Jr. V.M. Bove, editors, *SPIE Vol. 4518, Special session on Theoretical and Practical Issues in Digital Watermarking and Data*

Hiding, SPIE Multimedia Systems and Applications IV, pages 275–280, Denver, CO, 20-24 August 1998.

7. N. F. Johnson, Z. Duric, and S. Jajodia. Information hiding: Steganography and watermarking—attacks and countermeasures. In *Advances in Information Security 1*. Kluwer Academic Publishers, 2001.

8. E. Kawaguchi and R. O. Eason. The principle and applications of bpcs-steganography. In *SPIE International Symposium on Voice, Video, and Data Communications: Multimedia Systems and Applications*, pages 464–473, Boston, MA, November 2-4 1998.

9. E. Kawaguchi and M. Niimi. Modeling digital image into informative and noise-like regions by complexity measure. In *Information Modeling and Knowledge Bases IX*, pages 255–265. IOS Press, April 1998.

10. C. Kurak and J. McHugh. A cautionary note on image downgrading. In *Computer Security Applications Conference*, pages 153–159, San Antonio, Dec. 1992.

11. Y. Lee and L. Chen. An adaptive image steganographic model based on minimum-error lsb replacement. In *Ninth National Conference on Information Security*, pages 8–15, Taichung, Taiwan, 14-15 May 1999.

12. Y. Lee and L. Chen. A high capacity image steganographic model. In *IEE Vision, Image and Signal Processing*, 2000.

13. L. M. Marvel, C. G. Boncelet Jr., and C. T. Retter. Spread spectrum image steganography. *IEEE Trans. Image Processing*, 8:1075–1083, August 1999.

14. L.M. Marvel, G.W. Hartwig, and C. Boncelet. Compression-compatible fragile and semi-fragile tamper detection. In *SPIE EI Photonics West*, pages 131–139, San Jose, CA, 2000.

15. I. S. Moskowitz, L. Chang, and R. E. Newman. Capacity is the wrong paradigm. In *New Security Paradigms Workshop*, Virginia Beach, VA, USA, September 2002.

16. I. S. Moskowitz, N. F. Johnson, and M. Jacobs. A detection study of an NRL steganographic method. NRL Memorandum Report NRL/MR/5540–02-8635, Naval Research Laboratory, Code 5540, August 16 2002.

17. I. S. Moskowitz, G. E. Longdon, and L. Chang. A new paradigm hidden in steganography. In *New Security Paradigms Workshop*, pages 12–22, Ballycotton, County Cork, Ireland, Sept 2000. ACM (also appears in "The Privacy Papers" ed. R Herold, Auerbach Press 2002).

18. M. Niimi, H. Noda, and E. Kawaguchi. An image embedding in image by a complexity based region segmentation method. In *ICIP*, volume 3, pages 74–77, 1997.

19. W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.

20. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding – a survey. *Proceedings of the IEEE*, 87(7):1062–1078, July 1999.

21. N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, pages 323–335, August 2001.

22. N. Provos. Probabilistic methods for improving information hiding. Technical Report 01-1, CITI, University of Michigan, January 2001.

23. N. Provos and P. Honeyman. Detecting steganographic content on the internet. Technical Report 01-1, CITI, University of Michigan, August 2001.

24. G. Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, 1999.

25. A. Westfeld. F5 — a steganographic algorithm: High capacity despite better steganalysis. In *I.S. Moskowitz (Ed.) Information Hiding, LNCS 2137, IH 2001*, pages 289–302. Springer, 2001.

A Proof of Claim 4

Claim 4: Using the stego embedding described in subsection 5.3, a cover file I with at least K usable blocks can embed any self-delimited data string $s = s_1, s_2, ..., s_K$ correctly.

Proof of Claim 4: The sender tests each block of the cover image I in the π-permuted order $I_{\pi(1)}, I_{\pi(2)}, ...$ until K usable blocks have been found. Each usable block V_i encodes datum s_i by replacing it (if necessary) with block V_i' in the stego image I', and each unusable block U_i that comes before V_K in I is replaced (if necessary) with $U_i' \notin R$. The receiver tests each block of I' in the same order that the sender tests (and replaces if necessary) it, $I_1', I_2', ...$, until all of the embedded data $s_1', s_2', ..., s_K'$ have been decoded. We will prove that the string extracted by the receiver is the same as that embedded by the sender, assuming there is no noise in the transmission process, by induction on l, the number blocks of I' tested by the receiver.

Inductive Hypothesis: Let $n(l)$ be the number of usable blocks of I that occur in the first l blocks of I, that is, $V_{n(l)}$ is $I_{l'}$ for some $l' \leq l$, and $\forall l'', l' < l'' \leq l, I_{l''} = U_j$ for some j. For all $i < n(l-1)$, the ith decoded datum $s_i' = \Phi(V_i')$ is identical to the ith encoded datum, s_i.

Base Case: The base case, $i = 0$, is trivially true, and initially the decoded data string $s'[1..0]$ is empty, $s'[1..0] = \phi$.

Inductive Step: The inductive step will assume the hypothesis is true for $l-1$, and will show it to hold for l. Suppose that $l-1$ blocks of I have been tested, with $j = n(l-1)$ of them classified as rich (whose datum was extracted) and $l - j - 1$ of them classified as poor (and skipped). Then at this point the output data string is $s'[1..j] = s_1', s_2', ..., s_j'$, and by the inductive hypothesis, $\forall i \leq j, s_i' = s_i$.

The receiver then tests the next block $I_{\pi(l)}'$ to determine if it is rich.

If $I_{\pi(l)}' \in R$ then the receiver extracts datum $s_{j+1}' = \Phi(I_{\pi(l)}')$ and appends it to $s'[0..j]$ to produce $s'[0..j+1]$. $I_{\pi(l)}' \in R \Rightarrow I_{\pi(l)}' = V_{j+1}'$ since the sender leaves a rich block in I' before the end of s if and only if it encodes data, and the order in which the sender and receiver test and use blocks is the same. Thus $s_{j+1}' = \Phi(I_{\pi(l)}') = \Phi(V_{j+1}') = s_{j+1}$ and the inductive hypothesis holds for l.

Otherwise $I_{\pi(l)}'$ is poor, hence $I_{\pi(l)}'$ is U_{l-j}' and is skipped. This only happens before the end of s if the sender places a poor block $U_{l-j}' \notin R$ in I' that must be discarded by the receiver. In this case, the partially extracted string remains unchanged, and $n(l) = n(l-1) = j$ so the inductive hypothesis still holds for l.

If the block were rich and another datum were appended to $s'[0..j]$, the receiver tests $s'[0..j+1]$ to determine if it is complete (i.e., $\tau(s'[0..j+1]) = 1$ and the self-delimitation mechanism indicates that all of s has been extracted). If this is the case, then the receiver skips the rest of I' and outputs $s'[1..j+1] = s_1', s_2', ..., s_K' = s_1, s_2, ..., s_K$, since the inductive hypothesis holds for $l = K$ and no prefix of the self-delimiting data s tests true for completeness (i.e., $\forall i < K, \tau(s[1..i]) = 0$).