Mathematica Aeterna, Vol. 7, 2017, no. 4, 319 - 356

Information Hiding with Data Diffusion using Convolutional Encoding for Super-encryption

J M Blackledge
Stokes Professor, Science Foundation Ireland.
Director of Applied Research, Ministry of Defence, UK.
Honorary Professor, Dublin Institute of Technology, Ireland.
Distinguished Professor, Warsaw University of Technology, Poland.
Professor Extraordinaire, University of Western Cape, South Africa.

P Tobin
School of Electrical and Electronic Engineering,
Dublin Institute of Technology, Ireland.

J Myeza
Director of Libraries,
University of KwaZulu-Natal, South Africa.

C M Adolfo
Department of Systems Engineering,
Ministry of Defence, Military Technological College, Sultanate of Oman

Abstract

The unification of data encryption with information hiding methods continues to receive significant attention because of the importance of protecting encrypted information by making it covert. This is because one of the principal limitations in any cryptographic system is that encrypted data flags the potential importance of the data (i.e. the plaintext information that has been encrypted) possibly leading to the launch of an attack which may or may not be successful. Information hiding overcomes this limitation by making the data (which may be the plaintext or the encrypted plaintext) imperceptible, the security of the hidden information being compromised if and only if its existence is detected.

We consider two functions $f_1(\mathbf{r})$ and $f_2(\mathbf{r})$ for $\mathbf{r} \in \mathbb{R}^n$, $n = 1, 2, 3, ...$ and the problem of 'Diffusing' these functions together, applying a process we call 'Stochastic Diffusion' to the diffused field and then hiding the output of this process into one of the two functions. The coupling of these two processes using a form of conditioning that generates a well-posed inverse solution yields a super-encrypted field that is data-consistent.

After presenting the basic encryption method and (encrypted) information hiding model coupled with a mathematical analysis (within the context of 'convolutional encoding'), we provide a case study which is concerned with the implementation of the approach for full-colour 24-bit digital images. The ideas considered yields the foundations for a number of wide-ranging applications that include covert signal and image information interchange, data authentication, copyright protection and digital rights management. In this context, we also provide prototype software using m-code and Python for readers to use, improve upon and develop further for applications of interest.

Mathematics Subject Classification:
94A60, 94A08, 68P25, 11T71, 14G50, 60G35, 93E10.

Keywords:
Encryption, Steganography, Steganocryptography, Information Hiding, Data Diffusion, Stochastic Diffusion.

1 Introduction

Information or data hiding is the process of embedding and usually concealing data into similar or different forms of other data so that the hidden information is protected from unauthorised access [1] and embraces the principles associated with Watermarking and Steganography [2]. The term 'Information Hiding' can refer to either making the information imperceptible or keeping the existence of the information secret. In computer science it refers to the 'ability to prevent certain aspects of a software component from being accessible to its clients, using either programming language features (like private variables) or an explicit exporting policy', [3], [4].

The development of techniques for hiding information is important in situations when the use of data encryption is not feasible or when encryption cannot assure data security due to the encrypted information arousing suspicion or when the transmission of encrypted information is incriminating (e.g. in situations when the transmission of encrypted information is banned, is illegal or subject to investigatory powers [5]). Information hiding techniques embed plaintext data into covertext data that one wishes to send secretly via an innocuous 'message' based on the transmission of so-called stegotext data, which should be in a form that restricts detection or recovery of the hidden data. There are two principal data hiding categories, namely, Watermarking and Steganography [6].

Watermarking is the process of embedding information into another medium in a way that is difficult to remove which is useful to protect the source of the

information and avoid copyright violation, for example. An example of this concept is visible watermarking where the watermark is visible in the media used such as a text or logo which identifies the owner of the media. Other classes of watermarking include invisible watermarking, in which the information is added to the media in such a way that it cannot be recognised. Digital watermarking systems can be further categorised according to the robustness of an attack into fragile, semi-fragile and robust and can be used for copyright protection, source tracking or covert communications.

Steganography is the 'art' of embedding secret data into other data in such a way that no one, apart from the sender and intended recipient, suspects the existence of the secret data. The more recent use of this concept has emerged with the rapid development and communication of digital images, videos, and audio files which can be used as a medium for embedding important data. In other words, we currently live in a 'covertext rich environment' which is one of the principal reasons for the interest in, and, the development of, new steganographic techniques. The main advantage of Steganography is that messages do not attract attention to themselves and an examination of the data does not immediately reveal the existence of hidden information. Thus, with regard to a one-to-one communication protocol, the user sending the hidden data and the recipient of the data are the only 'users' who know about the existence of the hidden data.

Many steganographic techniques have been proposed, but steganography can be categorised into three basic types: (i) pure steganography in which the sender embeds the secret data directly, and the receiver extracts it likewise; (ii) private-key steganography where the sender uses a private key to embed the secret information in a way that is similar to private-key encryption; (iii) public-key steganography, in which the sender embeds the secret data using a private-key and the receiver extracts it using a public-key, a process that is similar to the public-key encryption.

Steganocryptography is a combined form of cryptography and steganography in which the data to be hidden is first encrypted before it is hidden in the covertext. This can pose certain restrictions on the way in which the encrypted data is hidden and requires that the extraction of the hidden cipher is achieved with minimal error so that an accurate decrypt can be obtained. This condition usually limits the robustness of the stegotext to transmission noise and other forms of distortion, i.e. the hidden encrypted data becomes fragile. On the other hand, it makes the stegotext tamper-proof, thereby giving the receiver evidence of an intercept and negating the potential for a decrypt to include disinformation that is taken by a recipient to be genuine. Most methods are divided into two categories; the first category focuses on embedding encrypted data in the spatial domain whereas the second category is based on the use of a transform domain to hide encrypted information. In this paper,

the latter approach is taken, the method developed being exclusively based on an application of the Fourier transform.

The approach reported in this paper considers a variation on the theme of convolutional encoding which involves two principal processes, namely, 'Data Diffusion' and 'Stochastic Diffusion', the latter method having been researched and implemented in a number of previous publications, e.g. [7], [8], [9], [10] and [11]. These processes are used to develop a highly fragile and thereby tamper-proof method of hiding encrypted data in a host data fields. The approach considered, which utilises the properties and characteristics of the Fourier transformation and the convolution and correlation integrals, is developed for arbitrary dimensions so that applications of the method can be used for encrypting and hiding information in digital signals, digital images and for three- and four-dimensional (i.e. three-dimensions + time) signal processing applications. We consider a case study that focuses on encrypted full-colour image information hiding with applications that can include image and e-document authentication, copyright protection and covert encryption for which prototype software is provided in Appendix A and Appendix B using m-code and Python, respectively.

To the best of the authors knowledge, the theoretical analysis presented in this paper (which is compounded in Theorem 3.1 given in Section 3.1) and the application reported in the Case Study given Section 5, is new and original, specifically, the coupling of the data diffusion and stochastic diffusion processes to produce an encryption scheme whose inverse is ill-posed in absence of the covertext.

2 Convolutional Coding

2.1 Context: Conventional Convolutional Encoders

Given a binary input stream $f[n] = \{0,1\}^{\ell}$ consisting of a 'block length' of ℓ bits (0 or 1), for a binary Finite Impulse Response function $g[n]$ say, a convolutional encoder yields an encoded output $h^{\ell}[n]$ given by [12]

$$h^{\ell}[n] = \sum_{m} g[n-m] f^{\ell}[m]$$

A convolutional encoder of this type describes a discrete linear time-invariant system which is a fundamental model for processing digital signals in general, when $f[n]$, $g[n]$ and $h[n]$ may be integer or floating point arrays.

Convolutional codes are used extensively to achieve reliable data transfer in numerous applications, such as digital video, radio, mobile communications and satellite communications [13]. These codes are often implemented in concatenation with a hard-decision code, and, prior to turbo codes, such construc-

tions were the most efficient, coming closest to the Shannon limit. One of the principal reason for using convolutional encoding is that maximum-likelihood decoding can be achieved with reasonable complexity using time-invariant trellis based decoders - the 'Viterbi algorithm' [14], which is a common example of an error correction code [15]. In this context, we consider an approach that is based on extending the convolutional encoding process to data fields of arbitrary dimensions while considering a modification to the process that yields a well-posed inverse solution to the decoding (deconvolution) problem.

2.2 Convolutional Encoding for $\mathbf{r} \in \mathbb{R}^n$

Consider a function $f(\mathbf{r})$ for $\mathbf{r} \in \mathbb{R}^n,\ n = 1, 2, 3, ...$ with n-dimensional Fourier and inverse Fourier transforms

$$F(\mathbf{k}) = \mathcal{F}_n[f(\mathbf{r})] \equiv \int_{-\infty}^{\infty} f(\mathbf{r}) \exp(-i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{r}$$

and

$$f(\mathbf{r}) = \mathcal{F}_n^{-1}[F(\mathbf{k})] \equiv \frac{1}{(2\pi)^n} \int_{-\infty}^{\infty} F(\mathbf{k}) \exp(i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{k}$$

respectively, $\mathbf{k}$ being the spatial frequency vector. If $g(\mathbf{r})$ is some stochastic function (a cipher) generated by a known algorithm or some other source (a 'code' or natural noise, for example), then convolutional encoding involves convolving $g(\mathbf{r})$ with $f(\mathbf{r})$ to produce an output $h(\mathbf{r})$ say, which we can write as ($\otimes$ denoting the n^{th} order convolution integral $\forall \mathbf{r} \in \mathbb{R}^n$)

$$h(\mathbf{r}) = g(\mathbf{r}) \otimes f(\mathbf{r}) \equiv \int_{-\infty}^{\infty} g(\mathbf{r} - \mathbf{r}') f(\mathbf{r}') d^n\mathbf{r}'$$

The transmission of such an output is taken to be corrupted by additive transmission noise described by the function $n(\mathbf{r})$, say, which introduces errors in to the recovery of $f(\mathbf{r})$ from $h(\mathbf{r})$ and thus we arrive at an equation of the form

$$h(\mathbf{r}) = g(\mathbf{r}) \otimes f(\mathbf{r}) + n(\mathbf{r}) \tag{1}$$

under the assumption that

$$\|n(\mathbf{r})\| << \|g(\mathbf{r}) \otimes f(\mathbf{r})\| \leq \|g(\mathbf{r})\| \times \|f(\mathbf{r})\|$$

the ratio $\|g(\mathbf{r}) \otimes f(\mathbf{r})\| / \|n(\mathbf{r})\|$ being known, in general, as the Signal-to-Noise Ratio or SNR in the fields of signal processing when $\mathbf{r} \in \mathbb{R}^1$ and image processing when $\mathbf{r} \in \mathbb{R}^2$.

2.3 Decoding: The Deconvolution Problem

This inverse (deconvolution) problem is as follows: Given equation (1), and, with functions $h(\mathbf{r})$ and $g(\mathbf{r})$ known, obtain a solution for $f(\mathbf{r})$. In some cases, $g(\mathbf{r})$ may not be known and the problem becomes the so-called 'blind deconvolution problem'. In general, this problem is an ill-posed problem, and consequently, has a range of solutions whose purpose is usually to regularise the inverse solution in such a way that an optimum estimate of $f(\mathbf{r})$ can be obtained subject to certain conditions. Most such solutions take the form

$$\hat{f}(\mathbf{r}) = q(\mathbf{r}) \otimes h(\mathbf{r}) \tag{2}$$

where $\hat{f}(\mathbf{r})$ is an estimate of $f(\mathbf{r})$, $q(\mathbf{r})$ is some filter with Fourier transform $Q(\mathbf{k}) = \mathcal{F}_n[q(\mathbf{r})]$. Well-examples include the following:

2.3.1 The Wiener Filter

$$\hat{f}(\mathbf{r}) = \mathcal{F}_n^{-1}[Q(\mathbf{k})H(\mathbf{k})]$$

where $H(k) = \mathcal{F}_n[h(\mathbf{r})]$,

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \mid N(\mathbf{k}) \mid^2 / \mid F(\mathbf{k}) \mid^2},\ G(\mathbf{k}) = \mathcal{F}_n[g(\mathbf{r})] \text{ and } N(\mathbf{k}) = \mathcal{F}_n[n(\mathbf{r})]$$

under the condition that $\|\hat{f}(\mathbf{r}) - f(\mathbf{r})\|_2^2$ is a minimum with the assumption that (signal independent noise) [16]

$$n(\mathbf{r}) \odot f(\mathbf{r}) = 0 \text{ and } f(\mathbf{r}) \odot n(\mathbf{r}) = 0,$$

$\odot$ being taken to denote the n^{th} order correlation integral $\forall \mathbf{r} \in \mathbb{R}^n$, i.e.

$$h(\mathbf{r}) = g(\mathbf{r}) \odot f(\mathbf{r}) \equiv \int_{-\infty}^{\infty} g(\mathbf{r} + \mathbf{r}')f(\mathbf{r}')d^n\mathbf{r}'$$

2.3.2 The Maximum Entropy Filter

$$\hat{f}(\mathbf{r}) = \exp\{-1 + 2\lambda[h(\mathbf{r}) \odot g(\mathbf{r}) - g(\mathbf{r}) \otimes \hat{f}(\mathbf{r}) \odot g(\mathbf{r})]\}$$

where λ is the Lagrange multiplier and is based on the Entropy, defined as

$$E = -\int_{-\infty}^{\infty} f(\mathbf{r}) \ln f(\mathbf{r})d^n\mathbf{r}$$

being a maximum, a 'solution' for $\hat{f}(\mathbf{r})$ that requires iteration to be applied. However, for $\lambda << 1$, we can write

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + 1/2\lambda}$$

which gives the linearised maximum entropy filter for estimate $\hat{f}(\mathbf{r})$ in the form of equation (2), [16].

2.3.3 The Maximum a Posteriori Filter

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \sigma_n^2/\sigma_f^2}$$

where σ_n and σ_f denote the Standard Deviations of the Gaussian Probability Density Functions $P[n(\mathbf{r})]$ and $P[f(\mathbf{r})]$ of $n(\mathbf{r})$ and $f(\mathbf{r})$, respectively, subject to the condition (Bayes rule) that, [16]

$$\frac{\partial}{\partial f} \ln P(h \mid f) + \frac{\partial}{\partial f} \ln P(f) = 0$$

2.3.4 Constrained Deconvolution

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \mid P(\mathbf{k}) \mid^2 / \lambda}$$

where $P(\mathbf{k}) = \mathcal{F}_n[p(\mathbf{r})]$ is any 'constraining spectrum' such that $\|p(\mathbf{r}) \otimes f(\mathbf{r})\|_2^2$ is a minimum, [16].

2.4 Decoding: The Deconvolution Problem for Special Functions

For certain functions $g(\mathbf{r})$ the deconvolution problem, as posed by equation (1), can be solved without resorting to a method of regularisation. The most important of these functions is the (unit amplitude) linear modulation function $g(\mathbf{r}) = \exp(i\alpha r^2),\ r \equiv \mid \mathbf{r} \mid$ for constant α. In this case, by correlating equation (1) with complex conjugate $g^*(\mathbf{r})$ (i.e. applying a 'matched filter') we obtain

$$g^*(\mathbf{r}) \odot h(\mathbf{r}) = \exp(-i\alpha r^2) \odot \exp(i\alpha r^2) \otimes f(\mathbf{r}) + \exp(-i\alpha r^2) \odot n(\mathbf{r})$$

$$= \delta^n(\mathbf{r}) \otimes f(\mathbf{r}) = f(\mathbf{r})$$

under the condition that $\exp(-i\alpha r^2) \odot n(\mathbf{r}) = 0$ (i.e. the stochastic function $n(\mathbf{r})$ does not correlate with the linear modulation function - the noise is 'signal independent') where δ^n is the n-dimensional Dirac delta function, [16]. This result is the basis for an approach to embedding information in signals and images with applications that include speech and document authentication, for example. It provides the basis for an information hiding model that yields the highest degree of resilience to distortion, especially with regard to additive noise (compared to other transformation-based information hiding methods) to known to date, [17] and [18]. In this context, a problem remains as to whether

a proof can be developed which proves that quadratic phase based impulse response functions of this type are unique in terms of their ability to code (and decode through correlation) information transmitted through high noise environments. If so, then it should be noted that such quadratic phase functions are generated by non-relativistic quantum mechanical systems in relation to the short-term transient behaviour of a beam of electrons, for example, propagating through free space upon the opening of a quantum shutter, [19]. This 'time scattering' effect could potentially be used for quantum (quadratic phase) information coding, a study of such an idea and the technology required lying beyond the scope of this publication.

2.5 Discussion

The nature of equation (1) and the conventional (discrete) convolutional encoding/decoding processes as discussed in this Section is based on a model for a n-dimensional signal that is ultimately related to the physical process that leads to the generation and detection of a signal, [20], [21]. Equation (1) is thus taken to be a fundamental model for the analysis and processing of signals in general (albeit for the linear and time or space invariant case) in a multitude of applications including acoustics and optics and other electromagnetic information systems. However, in regard to the development of data encryption methods, we are at liberty to 'invent' ideas that do not necessarily need to conform to a systems model derived from and constrained by the principles of physics, other than in the field of quantum cryptography. In this context, the foundations for the approach considered in this paper are compounded in the following section.

3 Generalised Well-posed Deconvolution

The ill-posed nature of the deconvolution problem compounded in equation (1) can lead to issues in the practical use of convolutional encoding in terms of providing a unique solution without the need to utilise error correction schemes and/or regularisation techniques as discussed in the previous section. We now consider a fundamental theorem associated with solving the deconvolution problem by 'designing' a problem that is well-posed.

3.1 Fundamental Theorem for Well-posed Deconvolution

The key to the original theme developed in this paper and applications thereof is compounded in the following theorem.

Theorem 3.1

For $\mathbf{r} \in \mathbb{R}^n,\ n = 1, 2, 3, ...$, given the equation,

$$h(\mathbf{r}) \otimes [g^*(\mathbf{r}) \odot g(\mathbf{r})] = f(\mathbf{r}) \otimes g(\mathbf{r}) \tag{3}$$

where $h(\mathbf{r})$ and $g(\mathbf{r})$ are known, the exact solution for $f(\mathbf{r})$ is given by

$$f(\mathbf{r}) = g^*(\mathbf{r}) \odot h(\mathbf{r}) \tag{4}$$

Proof of Theorem 3.1

Using the convolution and correlation theorems, equation (3) can be written in Fourier space as

$$H(\mathbf{k}) \mid G(\mathbf{k}) \mid^2 = F(\mathbf{k})G(\mathbf{k})$$

and multiplying both sides of this equation by $G^*(\mathbf{k})$ we have

$$G^*(\mathbf{k})H(\mathbf{k}) \mid G(\mathbf{k}) \mid^2 = G^*(\mathbf{k})F(\mathbf{k})G(\mathbf{k}) = F(\mathbf{k}) \mid G(\mathbf{k}) \mid^2$$

Hence,

$$F(\mathbf{k}) = G^*(\mathbf{k})H(\mathbf{k})$$

and using the correlation theorem, equation (4) is obtained.

Corollary 3.1

This result can be extended to include multiple convolutions of the function $g(\mathbf{r})$, since, if we consider the equation

$$h(\mathbf{r}) \otimes \prod_{j=1}^{\infty}{}_{\otimes} [g_j^*(\mathbf{r}) \odot g_j(\mathbf{r})] = f(\mathbf{r}) \otimes \prod_{j=1}^{\infty}{}_{\otimes} g_j(\mathbf{r})$$

where, by definition, for functions $q_1(\mathbf{r}),\ q_2(\mathbf{r}),\ ...$

$$\prod_{j=1}^{\infty}{}_{\otimes} q_j(\mathbf{r}) \equiv q_1(\mathbf{r}) \otimes q_2(\mathbf{r}) \otimes ...$$

then in Fourier space we have

$$H(\mathbf{k}) \prod_{j=1}^{\infty} \mid G_j(\mathbf{k}) \mid^2 = F(\mathbf{k}) \prod_{j=1}^{\infty} G_j(\mathbf{k})$$

and thus

$$\prod_{j=1}^{\infty} G_j^*(\mathbf{k})H(\mathbf{k}) \prod_{j=1}^{\infty} \mid G_j(\mathbf{k}) \mid^2 = F(\mathbf{k}) \prod_{j=1}^{\infty} \mid G_j(\mathbf{k}) \mid^2$$

giving the result

$$F(\mathbf{k}) = \prod_{j=1}^{\infty} G_j^*(\mathbf{k}) H(\mathbf{k})$$

or, after using the correlation theorem

$$f(\mathbf{r}) = \prod_{j=1}^{\infty}{}_{\odot} \; g_j^*(\mathbf{r}) \odot h(\mathbf{r})$$

where, by definition,

$$\prod_{j=1}^{\infty}{}_{\odot} \; q_j(\mathbf{r}) \equiv q_1(\mathbf{r}) \odot q_2(\mathbf{r}) \odot ...$$

Remark 3.1

Note that the solution for $f(\mathbf{r})$ does not rely on the condition $\mid G(\mathbf{k}) \mid^2 > 0 \; \forall \mathbf{k}$ (as is the case with the generalised deconvolution problem discussed in Section 2.3 if regularisation is not imposed) and is therefore a well-posed solution.

Remark 3.2

Equation (3) can be constructed thus,

$$h(\mathbf{r}) \otimes [g^*(\mathbf{r}) \odot g(\mathbf{r})] = g^*(\mathbf{r}) \odot f(\mathbf{r})$$

where upon the exect solution for $f(\mathbf{r})$ becomes

$$f(\mathbf{r}) = g(\mathbf{r}) \otimes h(\mathbf{r})$$

Remark 3.3

Equation (3) is consistent with an Infinite Impulse Response (IIR) filter model (which is common in the presence of a feedback topology) for a 'perfect feedforward filter'. The argument for this is as follows. For $\mathbf{r} \in \mathbb{R}^n, \; n = 1, 2, 3, ...$ the output from a IIR filter is given by

$$h(\mathbf{r}) = g(\mathbf{r}) \otimes f(\mathbf{r}) - p(\mathbf{r}) \otimes h(\mathbf{r})$$

where $g(\mathbf{r})$ denotes the feedforward filter function and $q(\mathbf{r})$ denotes the feedback filter function. Fourier transformation shows that

$$H(\mathbf{k}) = \frac{G(\mathbf{k}) Q^*(\mathbf{k})}{\mid Q(\mathbf{k}) \mid^2} F(\mathbf{k}), \; Q(\mathbf{k}) = 1 + P(\mathbf{k})$$

so that upon rearrangement and inverse Fourier transformation we obtain

$$h(\mathbf{r}) \otimes [q^*(\mathbf{r}) \odot q(\mathbf{r})] = g(\mathbf{r}) \otimes [q^*(\mathbf{r}) \odot f(\mathbf{r})]$$

Thus, in the case when the feedforward filter is a perfect filter, i.e. when $g(\mathbf{r}) = \delta^n(\mathbf{r})$, then

$$h(\mathbf{r}) \otimes [q^*(\mathbf{r}) \odot q(\mathbf{r})] = q^*(\mathbf{r}) \odot f(\mathbf{r})$$

and it is clear that $f(\mathbf{r})$ can be recovered from $h(\mathbf{r})$ by convolution with the feedback filter function, i.e.

$$f(\mathbf{r}) = q(\mathbf{r}) \otimes h(\mathbf{r}) = h(\mathbf{r}) + p(\mathbf{r}) \otimes h(\mathbf{r})$$

3.2 Connectivity with Einstein's Evolution Equation

Let $p(r),\ r \equiv \mid \mathbf{r} \mid$ denote the Probability Density Function (PDF) associated with the position in an n-dimensional space $\mathbf{r} \in \mathbb{R}^n$ where a particle can exist as a result of some 'random walk' generated by a sequence of 'elastic scattering' processes (with other like particles in a n-dimensional space, n =1,2 or 3), $p(r)$ being subject to the normalisation condition

$$\int_{-\infty}^{\infty} p(r)dr = 1$$

where $p(r)$ is a real function. Further, let $u(\mathbf{r}, t)$ denote the density function of a canonical assemble of particles all undergoing the same random walk process involving elastic scattering events (e.g. the number of particles per unit volume for the case when $n = 3$). Suppose we consider an infinite concentration of such particles at a time $t = 0$ located at an origin $\mathbf{r} = \mathbf{0}$ which can thereby be described by a perfect spatial impulse so that we can write $u(\mathbf{r}, 0) = \delta^n(\mathbf{r})$. The function which characterises the response of this system to such an impulse at a short time later $t = \tau << 1$ can then be taken to be given by

$$u(\mathbf{r}, \tau) = p(r) \otimes u(\mathbf{r}, 0) = p(r) \otimes \delta^n(\mathbf{r}) = p(r)$$

Thus, at any time t, the density field at some later time $t+\tau$, sourced by some 'source function' $s(\mathbf{r}, t)$, say, will be given by

$$u(\mathbf{r}, t + \tau) = p(r) \otimes u(\mathbf{r}, t) + s(\mathbf{r}, t) \tag{5}$$

This equation is Einstein's (multi-dimensional) evolution equation [22] and is a 'master equation' for elastic scattering processes in statistical mechanics, [23]. It assumes that the distribution $p(r)$ does not change with time so the

system is linear and stationary in a statistical sense. In this case there is a connectivity between equations (3) and (5) as shall now be shown.

If we apply a Taylor expansion in time to the function $u(\mathbf{r}, t+\tau)$, then it is clear that we can write equation (5) in the form

$$\tau\frac{\partial}{\partial t}u(\mathbf{r},t)+\frac{\tau^2}{2!}\frac{\partial^2}{\partial t^2}u(\mathbf{r},t)+...=-u(\mathbf{r},t)+u(\mathbf{r},t)\otimes p(r)+s(\mathbf{r},t)$$

By expressing the infinite series on the left hand side of the equation above in terms of some 'memory function' $m(t)$, say, we can write

$$\tau m(t)\otimes_t\frac{\partial}{\partial t}u(\mathbf{r},t)=-u(\mathbf{r},t)+u(\mathbf{r},t)\otimes p(r)+s(\mathbf{r},t) \tag{6}$$

where $\otimes_t$ is taken to denote the (causal) convolution integral over t. This is the inhomogeneous Generalised Kolmogorov-Feller Equation (GKFE), [24], [25], and, for any inverse function or class of inverse functions of the type $m^{-1}(t)$, say, such that

$$m^{-1}(t)\otimes_t m(t)=\delta(t)$$

can be written in the form, [26]

$$\tau\frac{\partial}{\partial t}u(\mathbf{r},t)=-m^{-1}(t)\otimes_t u(\mathbf{r},t)+m^{-1}(t)\otimes_t u(\mathbf{r},t)\otimes_{\mathbf{r}} p(r)+m^{-1}(t)\otimes_t s(\mathbf{r},t)$$

Let

$$f(\mathbf{r},t)=\tau m(t)\otimes_t\frac{\partial}{\partial t}u(\mathbf{r},t)-s(\mathbf{r},t)$$

so that we can write equation (6) as

$$p(r)\otimes u(\mathbf{r},t)-u(\mathbf{r},t)=f(\mathbf{r},t)$$

Fourier transforming into $\mathbf{k}$-space, application of the convolution theorem yields

$$U(\mathbf{k},t)[P(\mathbf{k})-1]=F(\mathbf{k},t)$$

where

$$U(\mathbf{k},t)=\mathcal{F}_n[u(\mathbf{r},t)],\ P(\mathbf{k},t)=\mathcal{F}_n[p(\mathbf{r})] \text{ and } F(\mathbf{k},t)=\mathcal{F}_n[f(\mathbf{r},t)]$$

which can be written in the form

$$U(\mathbf{k},t)\mid G(\mathbf{k})\mid^2=F(\mathbf{k},t)G^*(\mathbf{k}),\ G(\mathbf{k})=P(\mathbf{k})-1$$

Hence, upon inverse Fourier transformation, we obtain the equation

$$u(\mathbf{r},t)\otimes[g(\mathbf{r})\odot g(\mathbf{r})]=g(\mathbf{r})\odot f(\mathbf{r},t) \tag{7}$$

where

$$g(\mathbf{r}) = \mathcal{F}_n^{-1}[P(\mathbf{k}) - 1] = p(r) - \delta^n(\mathbf{r})$$

Comparing equation (7) with equation (3) it is clear that we have 'casted' Einstein's evolution equation (expressed in terms of the GKFE without loss of generality) in the prescribed form. Note that in the time independent case, equation (7) reduces to

$$u(\mathbf{r}) \otimes [g(\mathbf{r}) \odot g(\mathbf{r})] = -g(\mathbf{r}) \odot s(\mathbf{r})$$

showing that, via Theorem 3.1, we can derive an exact solution for the source function given knowledge of $u(\mathbf{r})$ and a model for $p(r)$, the result being given by

$$s(\mathbf{r}) = u(\mathbf{r}) - p(r) \otimes u(\mathbf{r})$$

which is, of course, compatible with the time independent form of equation (6). Further, we note that this equation is equivalent to Poisson's equation in the case when $p(r)$ is a Gaussian PDF with a small variance σ^2, i.e. for the Characteristic Function

$$P(k) = \int_{-\infty}^{\infty} p(r) \exp(-ikr) dr = \exp\left(-\frac{\sigma^2 k^2}{2}\right) \simeq 1 - \frac{\sigma^2 k^2}{2},\ \sigma^2 << 1,$$

$$S(\mathbf{k}) = U(\mathbf{k}) - P(k)U(\mathbf{k}) = \frac{\sigma^2 k^2}{2} U(\mathbf{k})$$

where $S(\mathbf{k}) = \mathcal{F}_n[s(\mathbf{r})]$ so that we can write

$$\nabla^2 u(\mathbf{r}) = -\frac{2}{\sigma^2} s(\mathbf{r})$$

which has the general Green's function solution, [27]

$$u(\mathbf{r}) = \frac{1}{2\pi\sigma^2 r} \otimes s(\mathbf{r}),\ \mathbf{r} \in \mathbb{R}^3;\ u(\mathbf{r}) = \frac{1}{\pi\sigma^2} \ln r \otimes s(\mathbf{r}),\ \mathbf{r} \in \mathbb{R}^2$$

4 Encrypted Information Hiding

Consider two functions $f_1(\mathbf{r})$ and $f_2(\mathbf{r})$ and the problem of how to hide an encrypted form of the function $f_1(\mathbf{r})$ (using convolutional encoding) by embedding it in the function $f_2(\mathbf{r})$ (or vice versa as required). The problem falls into the field *Steganocryptography*, cryptography being concerned with the encryption of information and steganography being concerned with the 'art' of hiding the content of one message in another. Here, we are interested in hiding encrypted information (a ciphertext in a covertext to produce an output known as a stegotext).

4.1 Data Hiding and Recovery

Ignoring the encryption process for now, consider the case when

$$\|f_1(\mathbf{r})\|_\infty = 1 \text{ and } \|f_2(\mathbf{r})\|_\infty = 1; \quad \|f(\mathbf{r})\|_\infty \equiv \sup\{|\ f(\mathbf{r})\ |: \mathbf{r} \in \mathbb{R}^n\}$$

and the additive 'information embedding equation'

$$h(\mathbf{r}) = cf_1(\mathbf{r}) + f_2(\mathbf{r}),\ c \in [0,1]$$

where the constant c is the 'Information Embedding Coefficient' (IEC). In this context $f_2(\mathbf{r})$ is referred to as the covertext, $h(\mathbf{r})$ is referred to as the stegotext and $f_1(\mathbf{r})$ is referred to as the plaintext, and, in order to hide the function $f_1(\mathbf{r})$ in $f_2(\mathbf{r})$ effectively, we required that

$$c\|f_1(\mathbf{r})\| << \|f_2(\mathbf{r})\|$$

Clearly, given that $\|f_1(\mathbf{r})\|_\infty = 1$ and $\|f_2(\mathbf{r})\|_\infty = 1$, the smaller the value of c, the smaller the perturbation of $f_1(\mathbf{r})$ to $f_2(\mathbf{r})$ becomes, and, in practical applications of the approach being considered, requires to be optimised in such a way that c is a minimum subject to optimal reconstruction $f_1(\mathbf{r})$ through application of the equation

$$f_1(\mathbf{r}) = \frac{1}{c}[h(\mathbf{r}) - f_2(\mathbf{r})]$$

We can also apply an information hiding strategy based on any exactly invertible transformation, e.g.

$$h(\mathbf{r}) = \mathcal{F}_n^{-1}[cF_1(\mathbf{k}) + F_2(\mathbf{k})],\ F_1(\mathbf{k}) = \mathcal{F}_n[f_1(\mathbf{r})],\ F_2(\mathbf{k}) = \mathcal{F}_n[f_2(\mathbf{r})]$$

and

$$f_1(\mathbf{r}) = \frac{1}{c}\mathcal{F}_n^{-1}[H(\mathbf{k}) - F_2(\mathbf{k})],\ H(\mathbf{k}) = \mathcal{F}_n[h(\mathbf{r})]$$

Also, note that since $\|f_1(\mathbf{r})\|_\infty = 1$, re-normalisation can be applied in the computation of $f_1(\mathbf{r})$ thereby eliminating the need to apply a multiplication by $1/c$, i.e. we can consider the equation

$$f_1(\mathbf{r}) = \frac{\mathcal{F}_n^{-1}[H(\mathbf{k}) - F_2(\mathbf{k})]}{\|\mathcal{F}_n^{-1}[H(\mathbf{k}) - F_2(\mathbf{k})]\|_\infty}$$

which avoids the need to know the precise value of c in the recovery of the hidden data $f_1(\mathbf{r})$ (as used in the designed of the function **Decrypt** discussed further in Section 5).

4.2 Data Diffusion

In order to recover the function $f_1(\mathbf{r})$ from $h(\mathbf{r})$ it is clear that $f_2(\mathbf{r})$ - the covertext - must be known. Because of this, we consider the process of 'diffusing' the two function $f_1(\mathbf{r})$ and $f_2(\mathbf{r})$ using convolutional encoding based on the properties of equation (3). Thus, consider the construction of the function

$$g(\mathbf{r}) = \mathcal{F}_n^{-1}\left[\frac{F_2(\mathbf{k})F_1(\mathbf{k})}{\mid F_2(\mathbf{k}) \mid^2}\right]$$

under the condition that if the power spectrum $\mid F_2(\mathbf{k}) \mid^2 = 0$ for any value of $\mathbf{k}$, then it is set to a value of 1 in order to avoid any singularities that may occur in the computation of the inverse filter $F_2(\mathbf{k})/ \mid F_2(\mathbf{k}) \mid^2$ given that in any and all cases when $F(\mathbf{k})_2 = 0$, $F_2(\mathbf{k})/ \mid F_2(\mathbf{k}) \mid^2 = F_2(\mathbf{k}) = 0$. Subject to this 'renormalisation condition', from Theorem 3.1, it is clear that $f_1(\mathbf{r})$ can be recovered from $g(\mathbf{r})$ by correlating $g(\mathbf{r})$ with $f_2^*(\mathbf{r})$. If we then apply the Fourier based hiding method discussed in the previous section, then we can hide the function $g(\mathbf{r})$ in the covertext function f_2 to construct the stegotext function as follows:

$$h(\mathbf{r}) = F_n^{-1}[cG(\mathbf{k}) + F_2(\mathbf{k})],\ G(\mathbf{k}) = \mathcal{F}_n[g(\mathbf{r})],\ c \in [0, 1]$$

4.3 Stochastic Diffusion

In addition to applying data diffusion we can go further and diffuse the function $g(\mathbf{r})$ with a stochastic function $s(\mathbf{r})$, say, which in practice, is taken to be generated by applying some key-dependent (cryptographically-strong) random number generating algorithm (with a uniform statistical distribution, a uniform power spectral density function, a high Lyapunov exponent and high cycle length, for example). We call this process 'Stochastic Diffusion' and is based on constructing the function

$$v(\mathbf{r}) = \mathcal{F}_n^{-1}\left[\frac{S(\mathbf{k})G(\mathbf{k})}{\mid S(\mathbf{k}) \mid^2}\right]$$

subject to the same re-normalisation condition as discussed in the previous section. We can then construct the following stegotext function

$$h(\mathbf{r}) = \mathcal{F}_n^{-1}[cV(\mathbf{k}) + F_2(\mathbf{k})],\ V(\mathbf{k}) = \mathcal{F}_n[v(\mathbf{r})],\ c \in [0, 1]$$

in the knowledge that, via Theorem 3.1, $f_1(\mathbf{r})$ can be recovered by correlating $v(\mathbf{r})$ with $s^*(\mathbf{r})$ to obtained $g(\mathbf{r})$ and then correlating $g(\mathbf{r})$ with $f_2^*(\mathbf{r})$ to recover the plaintext $f_1(\mathbf{r})$. Note that this process can be repeated, i.e. $v_1(\mathbf{r}) \equiv v(\mathbf{r})$ can be diffused with another stochastic function $s_2(\mathbf{r})$ to produce an output $v_2(\mathbf{r})$, the processing being repeated n times to produce function

$v_n(\mathbf{r})$ where each stochastic function $s_n(\mathbf{r})$ is assumed to have been generated by the same (or different for a multiple-algorithmic protocol) key-dependent pseudo-random number generating algorithm subject to a different key. This is because the convolution encoding process compounded in Theorem 3.1 is well-posed and has the repeating property compounded in Corollary 3.1 (see Section 3.1).

4.4 Steganalysis

The method of encryption and information hiding considered is compounded in the following Fourier space based equation:

$$H(\mathbf{k}) = F_2 + c\frac{S(\mathbf{k})}{\mid S(\mathbf{k}) \mid^2}\frac{F_2(\mathbf{k})}{\mid F_2(\mathbf{k}) \mid^2}F_1(\mathbf{k}) \tag{8}$$

where $H(\mathbf{k})$ coupled with equation (8) are known publicly (i.e. it is assumed that $h(\mathbf{r})$ can be intercepted and that the method of steganoencryption compounded in equation (8) is known) and $F_2(\mathbf{k})$ is known privately; in effect, $F_2(\mathbf{k})$ is a private key known only to the sender and recipient of the function $H(\mathbf{k})$ together with the algorithm used for generating the key-dependent stochastic field $s(\mathbf{r})$.

From equation (8) it is clear that there is one known and three unknown functions (ignoring the value of c) and the problem of recovering $F_1(\mathbf{k})$ from $H(\mathbf{k})$ is ill-posed given that a well-posed problem has the following properties: (i) a solution exists; (ii) the solution is unique; (iii) the behaviour of the solution changes continuously with the initial conditions.

Consider the case where the covertext $f_2(\mathbf{r})$ and thus $F_2(\mathbf{k})$ is known. In this case we can solve equation (8) and obtain a solution for $f_1(\mathbf{r})$ given by

$$f_1(\mathbf{r}) = s^*(\mathbf{r}) \odot w(\mathbf{r}) \tag{9}$$

where

$$w(\mathbf{r}) = \frac{1}{c}\mathcal{F}_n^{-1}[H(\mathbf{k})F_2^*(\mathbf{k}) - \mid F_2(\mathbf{k}) \mid^2]$$

and it is clear that can the plaintext $f_1(\mathbf{r})$ can now only be recovered, if and only if, the stochastic function $s(\mathbf{r})$ can be constructed. The problem is then reduced to the classic cryptanalysis problem, namely, given that the key-dependent algorithm for computing $s(\mathbf{r})$ is known, find the associated key(s). In this context, the application of a covertext function coupled with data diffusion prior to convolution based encryption provides a method for both hiding the cipher and enhancing the strength of the cipher given that the solution to equation (8) for $f_1(\mathbf{r})$ is ill-posed. Further, the diffusion of the function $f_1(\mathbf{r})$ with $f_2(\mathbf{r})$ prior to encryption (of the diffused field) provides a way of disguising any signature (statistical or otherwise) associated with cipher used.

Suppose that the plaintext function $f_1(\mathbf{r})$ happened to be known together with the covertext function $f_2(\mathbf{r})$ and the transmitted data $h(\mathbf{r})$; then from equation (9), it is clear that (using the correlation theorem) $s(\mathbf{r})$ can, in principle, be obtained from the equation

$$s(\mathbf{r}) = \mathcal{F}_n^{-1}\left[\frac{F_1^*(\mathbf{k})W^(\mathbf{k})}{\mid W(\mathbf{k}) \mid^2}\right],\ W(\mathbf{k}) = \mathcal{F}_n[w(\mathbf{r})]$$

But this result assumes that $\mid W(\mathbf{k}) \mid > 0 \forall \mathbf{k}$ illustrating that the de-correlation problem compounded in equation (9) is potentially ill-condition and therefore requires the application of the regularisation methods discussed in Section 2.3, for example, for which only a non-unique estimate to the stochastic function $s(\mathbf{r})$ can be found.

Assuming that the de-correlation problem posed by equation (9) can be solved (in terms of generating a conditional estimate), this result illustrates the importance of changing the key and/or algorithm for computing $s(\mathbf{r})$. For a single algorithm protocol (in which the pseudo-random number generating algorithm is used repeatedly - the more usual case) and a key generating algorithm based on the covertext alone (as considered in the Case Study - Section 5), the solution for $s(\mathbf{r})$ given above illustrates the importance of using a different covertext function for each stegotext transmission in order to minimise the potential for a successful attack. This requires a database of covertext function to be created and shared prior to application of the method proposed. Such a database would ideally be communicated using a one-time-pad and a personalised encryption engine as discussed in [30], [31] and [32], for example.

In general, it should be noted that steganalytic evidence is difficult to obtain unless a payload has been completely recovered and decrypted. In the absence of such a decrypt, only a statistical signature is available to indicate whether or not a file had been modified, a modification that may be the result of steganographic encoding or otherwise. Such a signature may only be of significance if the original covertext is known, at least in terms of current known attack strategies, pending the modification of Artificial Intelligence methods designed for detecting derived viruses [33] for the detection of encrypted payloads. Whether the approach considered can be shown to be fully resistant to both classical computers and quantum computers in the sense of the algorithms being secure against an attack by a quantum computer and cryptanalysis/steganalysis associated with post-quantum cryptography (e.g. [34], [35]) is an issue that lies beyond the scope of this publication, i.e. are the algorithms developed a form of quantum-resistant and quantum-safe encryption?

5 Case Study: Encrypted Image Information Hiding

Consider two digital images I_1 and I_2 of type real, each of which are regular matrices of size N×M whose elements are composed of floating point values between 0 and 1 inclusively (typically obtained by conversion to normalised floating point form from a k-bit image). Using the method discussed in Section 4, we now consider algorithms for encrypting/decrypting, hiding and recovering an image whose corresponding m-code is provided in Appendix A. For readers of this paper wishing to investigate the method discussed and extend it further using Python, the equivalent Python class is given in Appendix B.

The plaintext image I_1 is encrypted using both data and stochastic diffusion and the output hidden in covertext image I_2 generating a stegotext image I_3 from which a decrypt I_4 is then generated, all processes being applied separately to each of the RGB components of colour input images. Clearly, the differences between I_2 and I_3 should be a minimum as should the difference between I_1 and I_4 which is examined later.

The method assumes the use of floating point arithmetic throughout including writing the stegotext image to file. For this reason, a Tagged Image File Format is considered in which the floating point data is retained. This is a fundamental requirement in order to recover the data prior to application of the inverse processes required to output a decrypt which is highly sensitive to (floating point) errors introduced into the stegotext, thereby making the approach tamper-proof, i.e. floating point errors (subject to the floating point accuracy of the computations) introduced into the stegotext image through transmission noise or inspection by an attacker, including quantisation of the image, for example, leads to a erroneous decrypt.

5.1 Example

Figure 1 shows an example of the application of the m-code given in Appendix A, an overview of the design of this code being provided to accompany this example. Two test RGB colour images (with 8 bits per colour channel) each of size 1024×768 are used to illustrate the method for a value of $c = 10^{-4}$ (the IEC) whose estimate in terms of producing an optimal result is addressed in Section 5.2. The visual differences between I_2 and I_3 and between I_1 and I_4 are insignificant in terms of both colour and the grey-scale as presented in Figure 1, which is the result of a 24-bit colour to 8-bit grey-scale conversion. This visual insignificance is quantified numerically in Section 5.2. Diffusion of the images I_1 and I_2 is undertaken by function **Image_Diffusion** using the MATLAB Fast Fourier Transform algorithm **fft2**, the inverse process being accomplished using function **Inverse_Image_Diffusion**. In both cases, the

power spectrum is set to 1 if any spectral components are zero.

The stochastic function $s(\mathbf{r})$ is computed using the MATLAB function **rand** which returns a uniformly distributed matrix of pseudo-random floating point numbers of size N×M with floating point values between 0 and 1 inclusively. However, it is well known that functions such as MATLAB **rand**, which is based on 'Mersenne Twister', and conventional linear congruential methods of pseudo-random number generation are cryptographically weak. Thus, in 'field operations' of the method discussed, the **rand** function should be based on pseudo-random number generators that are known to be cryptographically strong, and, ideally, personalised algorithms using new classes of chaos-based algorithms obtained through the application of Evolutionary Computing and/or Artificial Intelligence, for example, [28] and [29].

The **rand** function is used in functions **Stochastic_Diffusion** which implements the convolution encoding process and **Inverse_Stochastic_Diffusion** which recovers the data using the same key set to the 'state' (the initial condition) of the pseudo-random number generator. While these keys can be generated independently by the user, because the covertext image is critical to computing the decrypt, in this example, we use the covertext to generate the keys directly. This is done by applying the equations (for each RGB colour component)

$$k_R = \lfloor \| aI_2^R \|_2 \rfloor,\ k_G = \lfloor \| aI_2^G \|_2 \rfloor \text{ and } k_B = \lfloor \| aI_2^B \|_2 \rfloor$$

where a is any large number whose magnitude determines the order of magnitude of the key length,

$$\| I \|_2 \equiv \sqrt{\sum_{n=1}^{N} \sum_{m}^{M} \mid I[n,m] \mid^2}$$

and $\lfloor x \rfloor \equiv \text{floor}(x)$ denotes an output that is the largest integer less than or equal to x.

It is envisaged that in the routine application of this algorithm, and, given that the keys used for stochastic diffusion are derived from the covertext, the sender and receiver of the stegotext would agree *a priori* upon a database of covertext images. Since the visual difference between the stegotext and covertext is insignificant, a visual inspection of the database using Thumbnails (i.e. reduced-size versions of the images contained in a database which serves the same role for images as a normal text index does for words as used by most modern operating systems or desktop and mobile environments) would be used to decrypt the encrypted image contained in the stegotext by the user choosing the image in the database that matches the received covertext. For large image databases, visual search engines could be used to produce a 'stegotext-covertext match'.

5.2 Numerical Analysis

The critical parameter affecting the performance of the encryption/decryption processes together with the covert extent of the data embedding processes (in terms of minimal distortion to the covertext by the perturbation of the embedded encrypted data through floating point addition) is the value of the IEC denoted by c.

Figure 1: Example application using the MATLAB Code given in Appendix A: Plaintext .bmp image (top-left) I_1, Covertext .bmp image (top-right) I_2, Stegotext .tiff image (bottom-left) I_3 and Decrypt .bmp image (bottom-right) I_4. Note that the images used, each of which are 1024×768 RGB colour images with 8 bits per colour channel, have been reproduced as 8-bit grey-level images for publication purposes only - full colour reproduction not being supported for this publication.

In order to evaluate the effects of changing the value of c, we consider the following Root Mean Square (RMS) error equations

$$e_1^R = \frac{1}{\sqrt{NM}}\|I_3^R - I_2^R\|_2,\ e_1^G = \frac{1}{\sqrt{NM}}\|I_3^G - I_2^G\|_2,\ e_1^B = \frac{1}{\sqrt{NM}}\|I_3^B - I_2^B\|_2$$

the total RMS RGB mean error being given by

$$e_1 = \frac{1}{3}\left(e_1^R + e_1^G + e_1^B\right) \tag{10}$$

and

$$e_2^R = \frac{1}{\sqrt{NM}}\|I_1^R - I_4^R\|_2,\ e_2^G = \frac{1}{\sqrt{NM}}\|I_1^G - I_4^G\|_2,\ e_2^B = \frac{1}{\sqrt{NM}}\|I_1^B - I_4^B\|_2$$

the total RMS RGB mean error being given by

$$e_2 = \frac{1}{3}\left(e_2^R + e_2^G + e_2^B\right) \tag{11}$$

Equations (10) and (11) are used to evaluate the (error) differences between

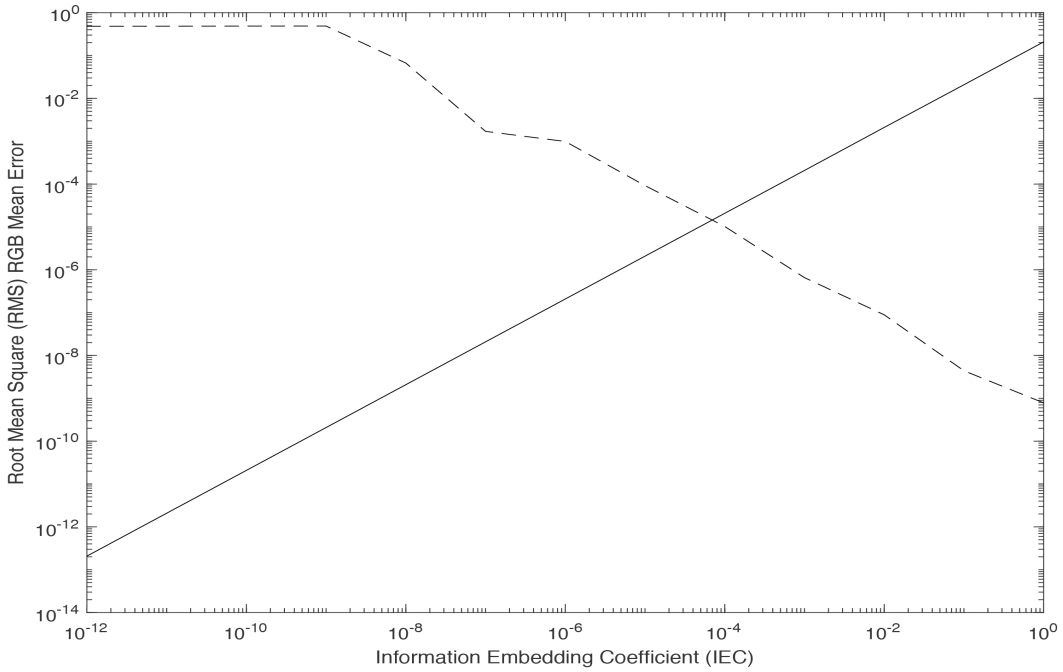

Figure 2: A log-log plot of the RMS RGB mean error (vertical axis) between the original covertext and the stegotext images (solid line), as defined by equation (10), and, the RMS RGB mean error (dashed line) between the original plaintext image and the reconstructed (decrypted) data, as defined by equation (11), for different values of the IEC (horizontal axis) between 1 and 10^{-12} in steps of 10^{-1} inclusively.

the covertext and the stegotext and between the plaintext and decrypt, respectively. The results are shown in Figure 2 which illustrates, as expected intuitively, that the smaller the value of the IEC, the smaller the RMS error

between the covertext and the stegotext but the greater the error between the plaintext and the decrypt. In the latter case, this is due to the effective floating point accuracy of the cipher when it is embedded in the covertext with a low value of the IEC. From Figure 2, we can consider an optimal value for the IEC to be the point of intersection between the two RMS 'error lines', i.e. at approximately $c = 10^{-4}$. However, it should be noted that for images of different sizes and types (e.g. colour images with different levels of quantisation and grey level images) and for computations conducted with different floating point accuracy, it is expected that the numerical performance of the process will yield different results to those given in Figure 2. Consequently, the optimal value of the IEC may change, further numerical analysis in this respect lying beyond the scope of this particular publication and being noted as a theme for a future investigation.

6 Software Development and Usage

Appendix A and Appendix B and provided to give readers access to source code that implements the algorithms discussed in this paper using m-code and Python, respectively. In both cases, copyright is attributed to J. M. Blackledge et al. and all rights are reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the organisation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The software listed in Appendices A and B is provided by the copyright holders and contributors *as is* and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright holders be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

7 Discussion and Conclusions

The use of personalised cryptographic algorithms coupled with the steganographic methods discussed in this paper and compounded in the application of Theorem 3.1, increases the threshold required for a successful attack to be launched in order to recover the plaintext. The attacker is first required to detect the existence of the encrypted information before an attempt can be made to decrypt it, the use of steganographic algorithms allowing for the existence of a ciphertext to be unknown. To recover the information, the attacker needs to first find a way of extracting the hidden encrypted information from the covertext and then decrypting it using the appropriate algorithm(s)/key(s). The exposure of the encryption key(s), the encryption algorithm(s) and the embedding technique to those other than the intended receiver is practically impossible provided, given the design of the key generating algorithm used in Section 5, the covertext is not compromised, and a different covertext is used for each transmission. In this context, greater security would be provided if the key(s), and, ideally the pseudo-random number generating algorithm used for stochastic diffusion, were generated independently from the covertext. This is of course at the cost of having to implement a separate key/algorithm-exchange protocol, but under the fundamental cryptographic principle: *One message, one key, one cipher*, and/or, in regard to the work reported here, *one covertext*.

The applications of the approach considered are numerous. Coupled with appropriate key-exchange protocols to initiate the use of cryptographically strong ciphers, the approach provides a generic method of encrypting and hiding high fidelity digital information, irrespective of the dimension of the data. The encrypted data is highly sensitive to transmission error and intolerant to distortion. The hidden data is therefore very fragile and hence, tamper-proof. This is due primarily to the method of information hiding which relies on floating point addition so that truncation of the stegotext due to quantisation involving transformation from floating point to integer form is not possible as is lossy compression, for example. In regard to digital signal applications involving audio files, for example, this is not an issue because audio files are composed of streams of floating point data (ignoring application specific data formatting and compression). However, digital images commonly rely on 'depth-quantisation' so that they can be displayed and retained as arrays composed of integers. This is why Least Significant Bit methods are so popular in image-based steganography, a method which has not been applied in this case. Thus, a further investigation that would be of value is to research different data embedding techniques, [36], other than the floating point additive approach considered here which necessitates the output image file having to be written in floating point form (i.e. as in function **Write_TIFF_Image** given in Appendix A).

Appendix A: Prototype MATLAB Code for Image Steganocryptography

The functions given in this Appendix have not been exhaustively tested and are provided to give the reader a quick guide to the basic software engineering required to implement the computational procedures discussed in Section 5, and, in turn, to help the reader appreciate the theoretical model developed in this paper. Where possible, the notation used for array variables and constants are based on the mathematical notation used in this paper or are acronyms for the function names. Note that the m-code given below has been somewhat condensed spatially in order to conform to the format of this publication while minimising the number of pages required to present it. The software was developed and implemented using (64-bit) MATLAB R2017b with double precision floating point arithmetic.

A1: Function to Encrypt Data

```
function Encrypt
%Read the colour images into arrays I1_C and I2_C, respectively
%noting that the mages are assumed to be of the same size.
I1_C=imread('Plaintext','bmp'); I2_C=imread('Covertext','bmp');
%Display plaintext and povertext images (as required).
figure(1), imshow(I1_C); figure(2), imshow(I2_C);
%Extract the RGB colour channels of both images.
I1_R =I1_C(:, :, 1); I1_G =I1_C(:, :, 2); I1_B =I1_C(:, :, 3);
I2_R =I2_C(:, :, 1); I2_G =I2_C(:, :, 2); I2_B =I2_C(:, :, 3);
[N,M]=size(I1_R);%Compute size of image arrays.
%Convert to normalised floating point form.
I1_R=im2double(I1_R); I1_G=im2double(I1_G); I1_B=im2double(I1_B);
I2_R=im2double(I2_R); I2_G=im2double(I2_G); I2_B=im2double(I2_B);
%Diffuse RGB components of the plaintext and covertext images,
ID_R=Image_Diffusion(I1_R,I2_R,N,M);
ID_G=Image_Diffusion(I1_G,I2_G,N,M);
ID_B=Image_Diffusion(I1_B,I2_B,N,M); %and generate RGB keys.
[Key_R,Key_G,Key_B]=Key_Generation(I2_R,I2_G,I2_B);
%Apply Stochastic Diffusion (modified convolution coding).
SD_R=Stochastic_Diffusion(ID_R,N,M,Key_R);
SD_G=Stochastic_Diffusion(ID_G,N,M,Key_G);
SD_B=Stochastic_Diffusion(ID_B,N,M,Key_B);
%Hide RGB components of data into RGB components of covertext
%using optimised value of information embedding coefficient c.
c=0.0001; I3_R=Hide_Data(I2_R,SD_R,c); I3_G=Hide_Data(I2_G,SD_G,c);
```

```
I3_B=Hide_Data(I2_B,SD_B,c);
%Reconstruct colour stegotext image and display (as required).
I = cat(3,I3_R,I3_G,I3_B); figure(3), imshow(I);
%Write stegotext image to file as Tagged Image File Format noting
%that it is critical the data be retained in floating point form.
Write_TIFF_Image(I);
```

A2: Function to Decrypt Data

```
function Decrypt
%Read covertext and stegotext images to arrays I2_C and I3_C
I2_C=imread('Covertext','bmp'); I3_C=imread('Stegotext.tif','tif');
%and display as required (both images taken to be of the same size).
figure(1), imshow(I2_C); figure(2), imshow(I3_C);
%Extract RGB components of the images converting the covertext into
%normalised floating point form and compute the size of the arrays.
I2_R =I2_C(:, :, 1); I2_G =I2_C(:, :, 2); I2_B =I2_C(:, :, 3);
I3_R =I3_C(:, :, 1); I3_G =I3_C(:, :, 2); I3_B =I3_C(:, :, 3);
I2_R=im2double(I2_R); I2_G=im2double(I2_G); I2_B=im2double(I2_B);
[N,M]=size(I2_R); %Recover the hidden (encrypted) data,
I1_R=Recover_Data(I2_R,I3_R); I1_G=Recover_Data(I2_G,I3_G);
I1_B=Recover_Data(I2_B,I3_B); %and regenerate RGB keys
[Key_R,Key_G,Key_B]=Key_Generation(I2_R,I2_G,I2_B);
%Decrypt data by application of Inverse Stochastic Diffusion
ISD_R=Inverse_Stochastic_Diffusion(I1_R,N,M,Key_R);
ISD_G=Inverse_Stochastic_Diffusion(I1_G,N,M,Key_G);
ISD_B=Inverse_Stochastic_Diffusion(I1_B,N,M,Key_B);
%Apply inverse image diffusion process
I1_R = Inverse_Image_Diffusion(ISD_R,I2_R);
I1_G = Inverse_Image_Diffusion(ISD_G,I2_G);
I1_B = Inverse_Image_Diffusion(ISD_B,I2_B);
%Combine RGB components to reconstruct the hidden (colour)
%image and display the result as required and write to file.
J = cat(3,I1_R,I1_G,I1_B); figure(3), imshow(J);
imwrite(J,'Decrypt.bmp','bmp');
```

A3: Common Functions

```
function [ID] = Image_Diffusion(I1,I2,N,M)
%Function to diffuse one image with another of the same size.
%Transform to Fourier space and compute the power spectrum P.
I1=fft2(I1); I2=fft2(I2); P=abs(I2).^2;
```

```
%Check to see if the power spectrum includes a 0, and,
%if so, set value of the power spectrum to 1.
for i=1:N
    for j=1:M
    temp=P(i,j); if temp==0 P(i,j)=1; else P(i,j)=P(i,j); end
    end
end
%Filter data.
ID=I2.*I1./P; %and Inverse Fourier transform, computing the
%real part of the data and normalise the result to give output.
ID=real(ifft2(ID)); ID=ID./max(max(ID));
```

```
function [I1] = Inverse_Image_Diffusion(ISD,I2)
%Transform the input data in to Fourier space.
ISD=fft2(ISD); I2=fft2(I2);
%Filter the data, apply inverse transformation,
%taking the real part and normalise output.
I1=conj(I2).*ISD; I1=real(ifft2(I1)); I1=I1./max(max(I1));
```

```
function [SD] = Stochastic_Diffusion(ID,N,M,Key)
%Compute array of random numbers determined by value of Key.
rand('state',Key); s=rand(N,M);
%Transform into Fourier space and compute power spectrum P.
ID=fft2(ID); S=fft2(s); P=abs(S).^2;
%Check to see if power spectrum includes 0 and if so set to 1.
for i=1:N
    for j=1:M
    temp=P(i,j); if temp==0 P(i,j)=1; else P(i,j)=P(i,j); end
    end
end %Filter the data, inverse Fourier transform and normalise.
SD=S.*ID./P; SD=real(ifft2(SD)); SD=SD./max(max(SD));
```

```
function [ISD] = Inverse_Stochastic_Diffusion(I1,N,M,Key)
%Convert input image into Fourier space.
I1=fft2(I1); %and regenerate pseudo-random number array for key.
rand('state',Key); s=rand(N,M); S=fft2(s);
%Filter the data, apply inverse Fourier transform and normalise.
ISD=conj(S).*I1; ISD=real(ifft2(ISD)); ISD=ISD./max(max(ISD));
```

```
function [Key_R,Key_G,Key_B] = Key_Generation(I2_R,I2_G,I2_B)
%Compute RGB keys by summing the arrays of the associated
%RGB components after multiplication by a large number whose
%value determines the order of magnitude of the key length,
%flooring results to nearest integers towards minus infinity.
I2_R=I2_R*1.0e+10; I2_G=I2_G*1.0e+10; I2_B=I2_B*1.0e+10;
Key_R=floor(sqrt(sum(sum(I2_R.*I2_R))));
Key_G=floor(sqrt(sum(sum(I2_G.*I2_G))));
Key_B=floor(sqrt(sum(sum(I2_B.*I2_B))));
```

```
function [I3] = Hide_Data(I2,SD,c)
%Hide encrypted image SD in covertext image I2 using Fourier
%space addition for information embedding coefficient c.
I3=real(ifft2(c*fft2(SD)+fft2(I2)));
```

```
function [I1] = Recover_Data(I2,I3)
%Subtract Fourier transform of covertext from
%Fourier transform of stegotext.
I1=real(ifft2(fft2(I3)-fft2(I2)));
```

```
function Write_TIFF_Image(I);
%Write image to TIFF file maintaining floating point values.
%Based on 'Writing an image with floating point values',
%Stackoverflow available at %https://stackoverflow.com/questions/
%14003402/writing-an-image-with-floating-point-values/33353930
t = Tiff('Stegotext.tif','w');
t.setTag('Photometric', Tiff.Photometric.RGB);
t.setTag('BitsPerSample', 64); t.setTag('SamplesPerPixel', 3);
tagstruct.RowsPerStrip = 16;
t.setTag('SampleFormat',Tiff.SampleFormat.IEEEFP);
t.setTag('ImageLength',size(I,1));
t.setTag('ImageWidth', size(I,2));
t.setTag('PlanarConfiguration', Tiff.PlanarConfiguration.Chunky);
tagstruct.Software = 'MATLAB'; t.setTag(tagstruct); t.write(I);
t.close();
```

Appendix B: Prototype Python Class and Methods for Image Steganocryptography

The software given in this Appendix has not been exhaustively tested and is provided to give the reader a Python class with methods which are, in effect,

a direct translation from the m-code given in Appendix A, thereby providing the reader with an open source of software that is independent of MATLAB. The code given below has been condensed spatially in order to conform to the format of this publication while minimising the number of pages required to present it. The software was developed and implemented using a 64-bit Unix environment with double precision floating point arithmetic. However, key generation is effectively based on 32-bit precision as the *Numpy* (uniformly distributed) pseudo-random number generator used - **rand** - can only be seeded with integer values between 0 and $2^{32} - 1$.

```
import numpy as np
from numpy.fft import fft2, ifft2
import tifffile

class StegCrypt(object):
  """
  Information Hiding with Data Diffusion using Convolutional
  Encoding for Super-encryption. Hides an image, the 'Plaintext',
  in a chosen camouflage image, the 'Covertext' to produce a
  'Stegotext' image. Also performs decryption of the stegotext
  image using the input covertext as a key.
  """

  # Hidden utility functions
  def _key_generator(self, channel):
    channel = channel * 1e10
    # 32-bit integer required for random seed in numpy
    key = int(np.floor(np.sqrt(np.sum(channel ** 2)))
              % 2 ** 32)
    return key

  def _hide_data(self, covertext, sdiffuse, c):
    return ifft2(c * fft2(sdiffuse) + fft2(covertext)).real

  def _recover_data(self, covertext, stegotext):
    return ifft2(fft2(stegotext) - fft2(covertext)).real

  def _image_diffusion(self, plaintext, covertext):
    plaintext = fft2(plaintext)
    covertext = fft2(covertext)
    p = np.abs(covertext ** 2)
    p[p == 0] = 1.
    diffuse = plaintext * covertext / p
```

```
  diffuse = ifft2(diffuse).real
  return diffuse / diffuse.max()

def _inverse_image_diffusion(self, diffuse, covertext):
  diffuse = fft2(diffuse)
  covertext = fft2(covertext)
  plaintext = covertext.conj() * diffuse
  plaintext = ifft2(plaintext).real
  return plaintext / plaintext.max()

def _stochastic_diffusion(self, diffuse, key):
  np.random.seed(key)
  arr_noise = fft2(np.random.rand(*diffuse.shape))
  p = np.abs(arr_noise ** 2)
  p[p == 0] = 1
  diffuse = fft2(diffuse)
  sdiffuse = diffuse * arr_noise / p
  sdiffuse = ifft2(sdiffuse).real
  return sdiffuse / sdiffuse.max()

def _inverse_stochastic_diffusion(self, sdiffuse, key):
  np.random.seed(key)
  noise = fft2(np.random.rand(*sdiffuse.shape))
  sdiffuse = fft2(sdiffuse)
  diffuse = noise.conj() * sdiffuse
  diffuse = ifft2(diffuse).real
  return diffuse / diffuse.max()

def encrypt(self, plaintext, covertext):
  """
  Hides the plaintext image within the provided covertext image

  Parameters
  ==========
  plaintext : array-like, shape (rows, columns, channels)
              The plaintext image to hide.
              Values must be ranging from 0 to 1

  covertext : array-like, shape (rows, columns, channels)
              The covertext image in which to hide the plaintext.
              Values must be ranging from 0 to 1
```

```
Returns
=======
stegotext : array-like, shape (rows, columns, channels)
                  The stegotext image

"""
c = 0.0001

if len(covertext.shape) != 2 and len(covertext.shape) != 3:
  raise Exception(ValueError, \
            "Input arrays must be 2- or 3-dimensional")

# Ensure inputs have the same shape
if not np.array_equal(covertext.shape, plaintext.shape):
  raise Exception(ValueError, \
           "Covertext and Plaintext shape do not match")

covertext_2D = False
if len(covertext.shape) == 2:
  covertext = covertext[:, :, None]
  plaintext = plaintext[:, :, None]
  covertext_2D = True

# Ensure images are 64-bit floating point
plaintext = plaintext.astype('float64')
covertext = covertext.astype('float64')
stegotext = np.zeros_like(covertext)

for i in range(plaintext.shape[-1]):

  plaintext_channel = plaintext[:, :, i]
  covertext_channel = covertext[:, :, i]
  key = self._key_generator(covertext_channel)

  # Hide each of the channels
  diff = self._image_diffusion(plaintext_channel,
                               covertext_channel)
  sdiff = self._stochastic_diffusion(diff, key)
  stegotext[:, :, i] = self._hide_data(covertext_channel,
                                       sdiff, c)

if covertext_2D:
```

```
    stegotext = stegotext[:, :, 0]

  return stegotext

def decrypt(self, stegotext, covertext):
  """
  Hides the plaintext image within the provided
  covertext image

  Parameters
  ==========
  stegotext : array-like, shape (rows, columns, channels)
              The stegotext image in which the plaintext
              image is hidden.

  covertext : array-like, shape (rows, columns, channels)
              The covertext image (the key)
              Values must be ranging from 0 to 1

  Returns
  =======
  plaintext : array-like, shape (rows, columns, channels)
              The hidden plaintext image

  """

  if len(covertext.shape) != 2 and len(covertext.shape) != 3:
    raise Exception(ValueError, \
             "Input arrays must be 2- or 3-dimensional")

  # Ensure inputs have the same shape
  if not np.array_equal(covertext.shape, stegotext.shape):
    raise Exception(ValueError, \
             "Covertext and Stegotext shape do not match")

  covertext_2D = False
  if len(covertext.shape) == 2:
    covertext = covertext[:, :, None]
    stegotext = stegotext[:, :, None]
    covertext_2D = True

  stegotext = stegotext.astype('float64')
```

```
    covertext = covertext.astype('float64')
    plaintext = np.zeros_like(stegotext)

    for i in range(stegotext.shape[-1]):

      covertext_channel = covertext[:, :, i]
      stegotext_channel = stegotext[:, :, i]
      key = self._key_generator(covertext_channel)

      # Recover the plaintext channel
      sdiff = self._recover_data(covertext_channel,\
                                 stegotext_channel)
      diff = self._inverse_stochastic_diffusion(sdiff, key)
      plaintext[:, :, i] = \
        self._inverse_image_diffusion(diff, covertext_channel)

    if covertext_2D == True:
      plaintext = plaintext[:, :, 0]

    return plaintext

  def save_stegotext_tiff(self, stegotext, filename):
    """
    Save stegotext as tiff file

    Parameters
    ==========
    stegotext : array-like, shape (rows, columns, channels)
                The stegotext to save

    filename : str
               The filename to save the stegotext

    Returns
    =======
    self : object

    """
    tifffile.imsave(filename, stegotext)
    return self

  def open_stegotext_tiff(self, filename):
```

```
    """
    Open a stegotext from a tiff file

    Parameters
    ==========
    filename : str
               The filename to of the stegotext image

    Returns
    =======
    stegotext : array-like, shape (rows, columns, channels)
                The stegotext array

    """
    stegotext = tifffile.imread(filename)
    if stegotext.dtype != 'float64':
      raise Exception(IOError, "Improperly saved stegotext file")
    return stegotext

  def open_tiff(self, filename):
    """
    Open an image from a tiff file
    Ensures that the tiff array has value (0, 1) after import.

    Parameters
    ==========
    filename : str
               The filename to of the tiff image

    Returns
    =======
    image : array-like, shape (rows, columns, channels)
            The normalised image array

    """
    tiff_image = tifffile.imread(filename).astype('float64')
    if tiff_image.max() > 1.0:
      tiff_image /= 255
    return tiff_image

# Unit tests
if __name__ == "__main__":
```

```
import matplotlib, os
matplotlib.use('tkagg')
import matplotlib.pyplot as pl

SC = StegCrypt()

# Produce a random field for the covertext and
# plaintext using a seeded rng
np.random.seed(3)
plaintext = np.random.rand(200, 200, 4)
covertext = np.random.rand(200, 200, 4)

# For each range of channels test the difference
# between the plaintext and its decrypted version
for i in [1, 2, 3, 4]:
  p = plaintext[:, :, 0:i]
  c = covertext[:, :, 0:i]
  s = SC.encrypt(p, c)
  p_d =  SC.decrypt(s, c)
  mean_difference = np.mean(np.abs(p - p_d))
  test = "FAIL"
  if mean_difference < 1e-3:
    test = "PASS"

  print "Test for channels 0-%d was %s with cost of %f" \
        %(i-1, test, mean_difference)


if os.path.exists("Plaintext.tiff") \
    and os.path.exists("Covertext.tiff"):
  # Now test on the image files
  plaintext = SC.open_tiff("Plaintext.tiff")[:, :, 0]
  covertext = SC.open_tiff("Covertext.tiff")[:, :, 0]
  stegotext = SC.encrypt(plaintext, covertext)

  # Save and re-open the stegotext using the
  # supplied functionality
  SC.save_stegotext_tiff(stegotext, "Stegotext.tiff")
  stegotext = SC.open_stegotext_tiff("Stegotext.tiff")

  # Decrypt the plaintext image
```

```
  plaintext_decrypt = SC.decrypt(stegotext, covertext)

  # Display the results
  f = pl.figure(figsize = (15, 5))
  ax = f.add_subplot(131)
  ax.imshow(plaintext)
  ax.set_title('Plaintext')
  pl.axis('off')
  ax = f.add_subplot(132)
  ax.imshow(covertext)
  ax.set_title('Covertext')
  pl.axis('off')
  ax = f.add_subplot(133)
  ax.imshow(plaintext_decrypt)
  ax.set_title('Decrypted Plaintext')
  pl.axis('off')
  pl.tight_layout()
  pl.show()

print "Unit tests finished"
```

Acknowledgements

The authors would like to acknowledge the support of Dublin Institute of Technology, the University of KwaZulu-Natal, the University of Western Cape and the Ministries of Defence in the UK and the Sultanate of Oman. It is acknowledged that the idea of coupling the process of data diffusion with that of stochastic diffusion for super-encryption was conceived by J M Blackledge, Director of Applied Research, Ministry of Defence, Military Technological College, Sultanate of Oman (2016-2017), who also developed the software provided in Appendix A and Appendix B.

References

[1] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, *Information Hiding: A Survey*, Proceedings of the IEEE (special issue), Vol. 87, No. 7, 1062-1078, (1999).

[2] I. Cox, M. Miller, J. Bloom, J. Fridrich and T. Kalker, *Digital Watermarking and Steganography*, Morgan-Kaufmann, 2007. (e-Book) ISBN: 9780080555805.

[3] *Information Hiding*, Wikipedia, the free Encyclopaedia. https://en.wikipedia.org/wiki/Information_hiding

[4] M. T. Raggo and C. Hosmer, *Data Hiding: Exposing Concealed Data in Multimedia, Operating Systems, Mobile Devices and Network Protocols*, Syngress, 2012, ISBN-13: 978-1597497435.

[5] The (UK) National Archives, *Regulation of Investigatory Powers Act 2000*, 2000. https://www.legislation.gov.uk/ukpga/2000/23/contents

[6] S. Katzenbeisser and F. A. Petitcolas, *Information Hiding: Techniques for Steganography and Digital Watermarking*, Artech House, Computer Security Series, 2000, ISBN: 1-58053-035-4.

[7] J. M. Blackledge and E. D. Coyle, *Information Hiding by Stochastic Diffusion and its Application to Printed Document Authentication*, Proc. of IET ISSC2009, UCD June 10-11, 2009, Vol. 20, No. 1, PS-4, 1-6, 2009.

[8] J. M. Blackledge, *Information Hiding using Stochastic Diffusion for the Covert Transmission of Encrypted Images*, Proc of IET ISSC2010 UCC Cork, 23-24 June, 2010.

[9] J. M. Blackledge and A. R. Al-Rawi, *Steganography using Stochastic Diffusion for the Covert Communication of Digital Images*, IANEG International Journal of Applied Mathematics, Vol. 41, Issue 4, 270-298, 2011.

[10] J. M. Blackledge and A. R. Al-Rawi, *Image Authentication using Stochastic Diffusion*, Systems Informatics: Modelling and Simulation, UKSIM2013, 10-12 April, Cambridge University, 2013.

[11] A. R. Al-Rawi, *Digital Rights Management using Steganocryptography*, PhD Thesis, Dublin Institute of Technology, 2013.

[12] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003, ISBN-13: 9780521642989. http://www.inference.org.uk/mackay/itila/

[13] R. Khanai and G. H. Kulkarni, *Performance Analysis of Conventional Crypto-coding*, International Journal of Latest Trends in Computing 193 Vol. 2, Issue 1, 193-197, 2011, e-ISSN: 2045-5364.

[14] *Viterbi algorithm*, Wikipedia, the free Encyclopaedia. https://en.wikipedia.org/wiki/Viterbi_algorithm

[15] R. Morelos-Zaragoza, *The Art of Error Correcting Codes*, Wiley, ISBN: 0470015586. http://the-art-of-ecc.com/

[16] J. M. Blackledge, *Digital Signal Processing: Mathematical and Computational Methods, Software Development and Applications*, Edition 2, Woodhead Publishing: Series in Electronic and Optical Materials, 2006; eBook ISBN: 9780857099457. https://arrow.dit.ie/engschelebk/4/

[17] J. M. Blackledge and O. Iakovenko, *On the Application of Two-dimensional Chirplets for Resilient Digital Image Watermarking*, Proc. IET ISSC2013, Letterkenny, Co Donegal, Ireland, June 20-21, 2013. http://toc.proceedings.com/20591webtoc.pdf

[18] J. M. Blackledge and O. Iakovenko, *Resilient Digital Image Watermarking for Document Authentication*, IAENG, International Journal of Computer Science, Vol. 41, No. 1, 1-17, 2014. http://www.iaeng.org/IJCS/issues_v41/issue_1/IJCS_41_1_01.pdf

[19] M. Mochinsky, *Transient Phenomena in Quantum Mechanics: Diffraction in Time* in *Paths of Discovery*, Pontifical Academy of Sciences, Acta 18, Vatican City 2006 http://www.pas.va/content/dam/accademia/pdf/acta18/acta18-moshinsky.pdf

[20] J. M. Blackledge, *Quantitative Coherent Imaging: Theory Methods and Applications*, Techniques in Physics, Academic Press, 1987; ISBN: 0-12-103300-7.

[21] J. M. Blackledge, *Digital Image Processing: Mathematical and Computational Methods*, Woodhead Publishing Series in Electronic and Optical Materials, 2005; ISBN-13: 978-1898563495. https://arrow.dit.ie/engschelebk/3/

[22] A. Einstein, *On the Motion of Small Particles Suspended in Liquids at Rest Required by the Molecular-Kinetic Theory of Heat*, Annalen der Physik, Vol. 17, 549-560, 1905.

[23] L. Navarro (communicated by J. Z. Buchwald), *Gibbs, Einstein and the Foundations of Statistical Mechanics* Arch. Hist. Exact Sci. 53 (1998) 147-180, Springer, 1998.

[24] A. N. Kolmogorov, *On Analytic Methods in Probability Theory*, in *Selected Works of A. N. Kolmogorov, Volume II: Probability Theory and Mathematical Statistics* (Ed. A. N. Shiryaev), Kluwer, Dordrecht, 61-108, 1992; From the Original: *Uber die Analytischen Methoden in der Wahrscheinlichkeitsrechnung*, 1931, *Math. Ann.* 104, 415-458, 1931.

[25] W. Feller, *On Boundaries and Lateral Conditions for the Kolmogorov Differential Equations*, *The Annals of Mathematics, Second Series*, Vol. 65, No. 3, 527-570, 1957.

[26] J. M. Blackledge and T. R. Rani, *Stochastic Modelling using Einstein's Evolution Equation*, International Journal for Pure and Applied Mathematics (Mathematica Aeterna), Vol. 7, No. 3, 193 - 210, 2017.

[27] G. Evans, J. M. Blackledge and P. Yardley, *Analytic Solutions to Partial Differential Equations*, Springer, 1999; ISBN: 2540761241.

[28] J. M. Blackledge, S. Bezobrazov, P. Tobin and F. Zamora, *Cryptography using Evolutionary Computing*, Proc. IET ISSC2013, Letterkenny, Co Donegal, Ireland, June 20-21, 2013.

[29] J. M. Blackledge, P. Tobin and S. Bezobrazov, *Cryptography using Artificial Intelligence*, The International Joint Con- ference on Neural Networks (IJCNN2015), Killarney, Ireland, 12-17 July, 2015.

[30] P. Tobin, L. Tobin, J. M. Blackledge and M. McKeever, *Chaos-based Cryptography for Cloud Computing*, IET ISSC2016, Ulster University, Derry, Northern Ireland, June 21-22, 2016.

[31] P. Tobin, L. Tobin, J. M. Blackledge and M. McKeever, *On the Development of a One-Time Pad Generator for Personalising Cloud Security*, Eighth International Conference on Cloud Computing, Grids and Virtualisation, Athens, Greece February 19 - 23, 2017.

[32] P. Tobin, L. Tobin, J. M. Blackledge and M. McKeever, *A Hardware One-Time Pad Prototype Generator for Localising Cloud Security*, 16th European Conference on Cyber Warfare and Security (ECCWS 2017), University College Dublin, Dublin June 29-30, 480-487, 2017.

[33] J. M. Blackledge, O. F. Asiru and M. T. Dlamini, *Application of Artificial Intelligence for Detecting Computing Derived Viruses*, 16th European Conference on Cyber Warfare and Security (ECCWS 2017), University College Dublin, Dublin June 29-30, 647-655, 2017.

[34] D. J. Bernstein, J. Buchmann and E. Dahmen (Eds.), *Post-Quantum Cryptography*, Springer, 2009; e-ISBN: 978-3-540-88702-7.

[35] D. J. Bernstein and T. Lange, *Post-Quantum Cryptography*, Nature, Vol. 549, 188-194, 2017.

[36] K. Saranya, C. S. Gnanadhas and M. George, *Data Embedding Techniques in Steganography*, International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 3, Issue 2, 200-205, 2013.

Received: November 11, 2017