2009-01-01

A Covert Encryption Method for Applications in Electronic Data Interchange

Jonathan Blackledge
Technological University Dublin, jonathan.blackledge@tudublin.ie

Dmitry Dubovitskiy
Oxford Recognition Limited, dda@oxreco.com

Recommended Citation

Blackledge, J., Dubovitskiy, D.: A Covert Encryption Method for Applications in Electronic Data Interchange. ISAST Journal on Electronics and Signal Processing, vol: 4, issue: 1, pages: 107 -128, 2009. doi:10.21427/D7RS67

A Covert Encryption Method for Applications in Electronic Data Interchange

Jonathan M Blackledge, Fellow, IET, Fellow, BCS and Dmitry A Dubovitskiy, Member IET

Abstract— **A principal weakness of all encryption systems is that the output data can be 'seen' to be encrypted. In other words, encrypted data provides a 'flag' on the potential value of the information that has been encrypted. In this paper, we provide a new approach to 'hiding' encrypted data in a digital image.**

In conventional (symmetric) encryption, the plaintext is usually represented as a binary stream and encrypted using an XOR type operation with a binary cipher. The algorithm used is ideally designed to: (i) generate a maximum entropy cipher so that there is no bias with regard to any bit; (ii) maximize diffusion in terms of key dependency so that a change in any bit of the key can effect any, and potentially all, bits of the cipher. In the work reported here, we consider an approach in which a binary or low-bit plaintext image is encrypted with a decimal integer or floating point cipher using a convolution operation and the output quantized into a 1-bit array generating a binary image ciphertext. This output is then 'embedded' in a host image to hide the encrypted information. Embedding is undertaken either in the lowest 1-bit layer or multiple 1-bit layers. Decryption is accomplished by: (i) extracting the binary image from the host image; (ii) correlating the result with the original cipher. In principle, any cipher generator can be used for this purpose and the method has been designed to operate with 24-bit colour images. The approach has a variety of applications and, in this paper, we focus on the authentication and self-authentication of e-documents (letters and certificates, for example) that are communicated over the Internet and are thereby vulnerable to attack (e.g. modification, editing, counterfeiting etc.). In addition to document authentication, the approach considered provides a way of propagating disinformation and a solution to scenarios that require 'plausible deniability'.

Index Terms— **Covert encryption, Steganography, Information hiding, Authentication**

I. Introduction

One of the principal weaknesses of all encryption systems is that the form of the output data (the ciphertext), if intercepted, alerts the intruder to the fact that the information being transmitted may have some importance and that it is therefore worth attacking and attempting to decrypt it. This aspect of ciphertext transmission can be used to propagate disinformation, achieved by encrypting information that is specifically designed to be intercepted and decrypted. In this case, we assume that the intercept will be attacked, decrypted and the information retrieved. The 'key' to this approach is to make sure that the ciphertext is relatively strong (but not too strong!) and that the information extracted is of good quality in terms of providing the attacker with 'intelligence' that is perceived to be valuable and compatible with their expectations, i.e. information that reflects the concerns/interests of the individual(s) and/or organisation(s) that encrypted the data. This approach provides the interceptor with a 'honey pot' designed to maximize their confidence especially when they have had to put a significant amount of work in to 'extracting it'. The trick is to make sure that this process is not too hard or too easy. 'Too hard' will defeat the object of the exercise as the attacker might give up; 'too easy', and the attacker will suspect a set-up!

In addition to providing an attacker with a honey-pot for the dissemination of disinformation, it is of significant value if a method can be found that allows the real information to be transmitted by embedding it in non-sensitive information after (or otherwise) it has been encrypted, e.g. camouflaging the ciphertext. This is known as *Steganography* which is concerned with developing methods of writing hidden messages in such a way that no one, apart from the intended recipient, knows of the existence of the message in contrast to cryptography in which the existence of the message itself is not disguised but the content is scrambled [1], [2]. Steganography provides a significant advantage over cryptography alone in that messages do not attract attention to themselves, to messengers, or to recipients. No matter how well plaintext is encrypted (i.e. how unbreakable it is), by default, a ciphertext will arouse suspicion and may in itself be incriminating, as in some countries encryption is illegal.

This paper presents a method of 'hiding' encrypted information in a colour digital image. In principle, any cipher can be used to do this providing it consists of floating point numbers that are ideally, uniformly distributed. The scheme allows for the authentication and self-authentication of documents such as letters, certificates and other image based data. The encrypted watermark can be camouflaged to obfuscate its existence and the applications to which the method can be applied are numerous. For example, the self-authentication of e-documents sent as attachments over the internet provides a unique facility for many legal and financial transactions that have traditionally relied on paper based documents to secure authenticity. The method also provides a unique way of 'propagating' disinformation in the form of an encrypted document which contains hidden information.

In order to introduce the background to the approach reported in this paper and to set it in a wider context, Sections II and III provide an overview of Steganography and some of the

Manuscript completed in August, 2009. The work reported in this paper is supported by the Science Foundation Ireland.

Jonathan Blackledge (email: jonathan.blackledge@dit.ie) is SFI (Science Foundation Ireland) Stokes Professor, School of Electrical Engineering Systems, Faculty of Engineering, Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland - http://eleceng.dit.ie/blackledge. Dr Dmitry Dubovitskiy is Director of Oxford Recognition Limited (email: dda@oxreco.com).

management issues associated with data security technology. This material is designed to place the principle of 'covert encryption' in the category of applications for which it is designed.

II. Steganography

The word 'Steganography' is of Greek origin and means 'covered', or 'hidden writing'. In general, a steganographic message appears as something else known as a covertext. By way of a simple illustrative example, suppose we want to transmit the phrase

```
The Queen likes horses
```

which is encrypted to produce the cipher stream

```
syoahfsuyTebhsiaulemNG
```

This is clearly a scrambled version of a message with no apparent meaning to the order of the letters from which it is composed. Thus, it is typical of an intercept that might be attacked because of the very nature of its incomprehensibility. However, suppose that the cipher stream above could be recast to produce the phrase

```
Beware of Greeks bearing gifts
```

If this phrase is intercepted it may not be immediately obvious that there is alternative information associated with such an apparently innocuous message, i.e. if intercepted, it is not clear whether or not it is worth initiating an attack.

The conversion of a ciphertext to another plaintext form is called *Stegotext* conversion and is based on the use of *Covertext*. Some covertext must first be invented and the ciphertext mapped on to it in some way to produce the stegotext. This can involve the use of any attribute that is readily available such as letter size, spacing, typeface, or other characteristics of a covertext, manipulated in such a way as to carry a hidden message. The basic principle is given below:

$$\begin{array}{ccccc} & & \text{Data} & \rightarrow & \text{Covertext} \\ & & & & \downarrow \\ \text{Plaintext} & \rightarrow & \text{Ciphertext} & \rightarrow & \text{Stegotext} \\ & & & & \downarrow \\ & & & & \text{Transmission} \end{array}$$

Note that this approach does not necessarily require the use of plaintext to ciphertext conversion as illustrated above and that plaintext can be converted into stegotext directly. A simple approach to this is to use a mask to delete all characters in a message except those that are to be read by the recipient of the message. For example, consider the following message:

```
At what time should I confirm our
activities? kindly acknowledge.
```

This seemingly innocent plaintext could be used to hide the message

```
Attack now
```

through application of the following mask:

```
11000010000000000000000000000000000
```

```
11000000000010000010001110000000
```

where 0 denotes that a character or space is to be ignored and 1 denotes that a character or space is used. Apart from establishing a method of exchanging the mask which is equivalent to the key in cryptography, the principal problem with this approach is that different messages have to be continuously 'invented' in order to accommodate hidden messages and that these 'inventions' must appear to be legitimate statements. However, the wealth of data that is generated and transmitted in todays environment and the wide variety of formats that are used means that there is much greater potential for exploiting steganographic methods than were available before the 'IT revolution'. In other words, the IT revolution has generated a camouflage rich environment in which to operate and one can attempt to hide plaintext or ciphertext (or both) in a host of data types, including audio and video files and digital images. Moreover, by understanding the characteristics of a transmission environment, it is possible to conceive techniques in which information can be embedded in the transmission noise, i.e. where natural transmission noise is the covertext. There are some counter measures - *steganalysis* - that can be implemented in order to detect stegotext. However the technique usually requires access to the covertext which is then compared with the stegotext to see if any modifications have been introduced. The problem is to find ways of obtaining the original covertext.

A. Hiding Data in Images

The relatively large amount of data contained in digital images makes them a useful 'medium' for undertaking steganography. Consequently digital images can be used to hide messages in other images. A colour image typically has 8 bits to represent the Red, Green and Blue components. Each colour component is composed of 256 'colour values' (for a 24-bit image) and the modification of some of these values in order to hide other data is undetectable by the human eye. This modification is often undertaken by changing the least significant bit in the binary representation of a colour or grey level value (for grey level digital images). For example, for 7-bit ASCII conversion, the grey level value 100 has the binary representation 1100100 which is equivalent to the character d. If we change the least significant bit to give 1100101 (which corresponds to a grey level value of 101 and character e) then the difference in the output image will not be discernable even though we have replaced the letter e with default character d. In this way, the least significant bit can be used to encode information other than pixel intensity and the larger the host image compared with the hidden message, the more difficult it is to detect the message. In this way it is possible to hide an image in another image for which there are a number of approaches available (including the application of bit modification). For example, Figure 1 shows the effect of hiding one image in another through the process of re-quantization and addition. The image to be embedded is re-quantized to just 3-bits or 8 grey levels so that it consists of an array of integer values between 0 to 7. The result is then added to the host image (an array of values between 0 and

255) on a pixel by pixel basis such that if the output exceeds 255 then it is truncated (i.e. set to 255). The resulting output is slightly brighter with minor distortions in some regions of the image that are homogeneous.

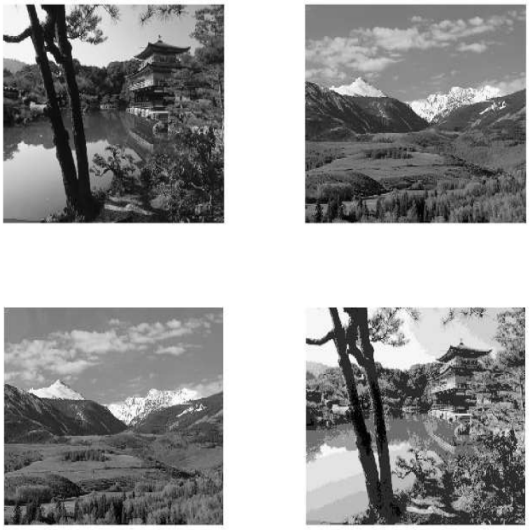

Fig. 1. Illustration of 'hiding' one image (top left) in another image (top right) through simple re-quantization and addition (bottom left). By subtracting the bottom left image from the top right image and re-quantizing the output, the bottom right reconstruction is obtained.

Clearly, knowledge of the original host image allows the hidden image to be recovered (by subtraction) giving a result that is effectively completely black. However, by increasing its brightness, the hidden image can be recovered as shown in Figure 1 which, in this example, has been achieved by re-quantizing the data from 0-7 back to 0-255 grey levels. The fidelity of this reconstruction is poor compared to the original image but it still conveys the basic information, information that could be covertly transmitted through the host image as an email attachment, for example. Note that the host image represents, quite literally, the key to recovering the hidden image. The additive process that has been applied is equivalent to the 'process of confusion' that is the basis for a substitution cipher. Rather than the key being used to generate a random number stream using a pre-defined algorithm from which the stream can be re-generated (for the same key), the digital image is, in effect, being used as the cipher. Note that the distortion generated by re-quantization means that the same method can not be used if the hidden image is encrypted. The degradation in the ciphertext will not allow an accurate decrypt to be accomplished due to loss of data.

Steganography is often used for digital watermarking. This is where the plaintext, which acts as a simple identifier containing information such as ownership, copyright and so on, is hidden in an image so that its source can be tracked or verified. This is equivalent to hiding a 1-bit image in a host image as illustrated in Figure 2 which uses the same method as discussed above. In this example, a columnar transposition cipher [3], [4] has been used to encrypt this sentence using the keyword: *Steganography* - see Appendix I. The grid is given by

```
11 12 03 04 01 07 08 05 10 02 09 06 13
I  n     t  h  i  s     e  x  a  m  p
l  e  ,     a     c  o  l  u  m  n  a
r     t  r  a  n  s  p  o  s  i  t  i
o  n     c  i  p  h  e  r     h  a  s
b  e  e  n     u  s  e  d     t  o
e  n  c  r  y  p  t     t  h  i  s
s  e  n  t  a  n  c  e     u  s  i  n
g     t  h  e     k  e  y  w  o  r  d
:     S  t  e  g  a  n  o  g  r  a  p
h  y
```

and the ciphertext is

```
haai yaeexus  huwg ,t ecnts
t rcnrtht opee eenmntaosira
i npupn gscshstckaamihtisor
elordt yoIlrobesg:hne nene
 ypais  ndp
```

Note that the host image is required to recover the ciphertext information and is thus the 'key' to this (steganographic) process.

The methods discussed above refer to electronic-to-electronic type communications in which there is no loss of information (assuming the image is not compressed to JPEG - Joint Photographics Expert Group - format, for example). Steganography and watermarking techniques can be developed for hardcopy data which have a range of applications. However, these techniques have to be robust to the significant distortions generated by the printing and scanning processes. A simple approach is to add information to a printed page that is difficult to see. For example, some modern colour laser printers, including those manufactured by HP and Xerox, print tiny yellow dots which are added to each page. The dots are barely visible and contain encoded printer serial numbers, date and time stamps. This facility provides a useful forensics tool for tracking the origins of a printed document.

Fig. 2. Binary image of encrypted information (right), obtained by subtraction of the covertext image from the stegotext image (left).

B. Hiding Data in Noise

The art of steganography is to use what ever covertext is readily available to make the detection of plaintext or, ideally, the ciphertext as difficult as possible. This means that the embedding method used to introduce the plaintext/cipherext into the covertext should produce a stegotext that is indistinguishable from the covertext in terms of its statistical characteristics

and/or the information it conveys. From an information theoretic point of view, this means that the covertext should have significantly more capacity than the cipheretext, i.e. there must be a high level of redundancy. Utilising noisy environments often provides an effective solution to this problem. There are three approaches that can be considered:

- embedding the ciphertext in real noise;
- transforming the ciphertext into noise that is then added to data;
- replacing real noise with ciphertext that has been transformed in to synthetic noise with exactly the same properties as the real noise.

In the first case we can make use of noise sources such as thermal noise, flicker noise, and shot noise associated with electronics that digitize an analogue signal. In digital imaging this may be noise from the imaging charge couple device (CCD) element; for digital audio, it may be noise associated with the recording techniques used or amplification equipment. Natural noise generated in electronic equipment usually provides enough variation in the captured digital information that it can be exploited as a noise source to 'cover' hidden data. Because such noise is usually a linear combination of different noise types generated by different physical mechanisms, it is usually characterised by a normal or Gaussian distribution as a result of the Central Limit Theorem.

In the second case, the ciphertext is transformed into noise whose properties are consistent with the noise that is to be expected in certain data fields. For example, lossy compression schemes (such as JPEG) always introduce some error (numerical error) into the decompressed data and this can be exploited for steganographic purposes. By taking a clean image and adding ciphertext noise to it, information can be transmitted covertly providing all users of the image assume that it is the output of a JPEG or some other lossy compressor. Of course, if such an image is JPEG compressed, then the covert information may be badly corrupted.

In the third case, we are required to analyse real noise and derive an algorithm for its synthesis. Here, the noise has to be carefully synthesized because it may be readily observable as it represents the data stream in its entirety rather than data that is 'cloaked' in natural noise. This technique also requires that the reconstruction/decryption method is robust in the presence of real noise that is assumed will be added to the synthesized noise during a transmission phase. In this case, random fractal models are of value because the spectral properties of many noise types found in nature signify fractal properties to a good approximation [5]. This includes transmission noise over a range of radio and microwave spectra, for example, and Internet traffic noise.

With regard to Internet traffic noise the time series data representing packet size and inter-arrival times shows well defined random fractal properties [6]. There are a range of time-series models that can be used to characterise Internet traffic noise based on the number of packets (or bytes) as a function of time. Fractal time-series models are applicable when the underlying processes have a similar appearance regardless of the time scale over which they are observed.

An example of the random fractal nature of Internet traffic is given in Figure 3 which shows the number of packets per unit time over four different time scales. Compared with a time-series model based on Poisson statistics (i.e. a Poisson random number generator) as shown in Figure 3, it is clear that Internet traffic noise has fractal characteristics (i.e. has self-affine behaviour). Like real traffic, Internet traffic changes over a daily cycle according to the number of users. However, much of the traffic 'riding' the Internet can be modelled using fractals and as the Internet has become larger and larger, the fractal nature of the traffic has become more and more pronounced.

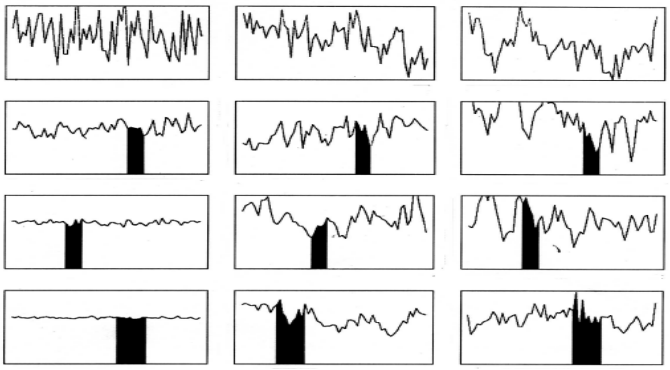

Fig. 3. Simulated Poisson based time-series model (left), real Internet traffic time-series (centre) and simulated fractal time-series model taken over four different time scales (from top to bottom respectively). The shaded areas highlight the data displayed in the plot above, respectively.

C. Disinformation

Disinformation is used to tempt the enemy into believing certain kinds of information. The information may not be true or contain aspects that are designed to cause the enemy to react in an identifiable way that provides a strategic advantage [7], [8]. Camouflage, for example, is a simple example of disinformation [9]. This includes techniques for transforming encrypted data into forms that resemble the environments through which an encrypted message is to be sent. At a more sophisticated level, disinformation can include encrypted messages that are created with the sole purpose of being broken in order to reveal information that the enemy will react to by design.

Disinformation includes arranging events and processes that are designed to protect against an enemy acquiring knowledge of a successful encryption technology and/or a successful attack strategy. A historically significant example of this involved the Battle of Crete which began on the morning of 20 May 1941 when Nazi Germany launched an airborne invasion of Crete under the code-name Unternehmen Merkur (operation mercury) [10]. During the next day, through miscommunication and the failure of Allied commanders to grasp the situation, the Maleme airfield in western Crete fell to the Germans which enabled them to fly in heavy reinforcements and overwhelm the Allied forces. This battle was unique in two respects: (i) it was the first airborne invasion

in history[1]; (ii) it was the first time the Allies made significant use of their ability to read Enigma codes. The British had known for some weeks prior to the invasion of Crete that an invasion was likely because of the work being undertaken at Bletchley Park. They faced a problem because of this. If Crete was reinforced in order to repel the invasion then Germany would suspect that their encrypted communications were being compromised. But this would also be the case if the British and other Allied troops stationed on Crete were evacuated. The decision was therefore taken by Churchill to let the German invasion proceed with success but not without giving the invaders a 'bloody nose'. Indeed, in light of the heavy casualties suffered by the parachutists, Hitler forbade further airborne operations and Crete was dubbed 'the graveyard of the German parachutists'. The graveyard for German, British, Greek and Allied soldiers alike was not a product of a fight over desirable and strategically important territory (at least for the British). It was a product of the need to secure Churchill's 'Ultra-secret'. In other words, the Allied efforts to repulse the German invasion of Crete were, in reality, a form of disinformation, designed to secure a secret that was, in the bigger picture, more important than the estimated 16,800 dead and wounded that the battle cost.

D. Plausible Deniability

Deniable encryption allows an encrypted message to be decrypted in such a way that different and plausible plaintexts can be obtained using different keys [11]. The idea is to make it impossible for an attacker to prove the existence of the real message, a message that requires a specific key. This approach provides the user with a solution to the 'gun to the head problem' as it allows the sender to have plausible deniability if compelled to give up the encryption key. There are a range of different methods that can be designed to implement such a scheme. For example, a single ciphertext can be generated that is composed of randomised segments or blocks of data which correlate to blocks of different plaintexts encrypted using different keys. A further key is then required to assemble the appropriate blocks in order to generate the desired decrypt. This approach, however, leads to ciphertext files that are significantly larger than the plaintexts they contain. On the other hand, a ciphertext file should not necessarily be the same size as the plaintext file and padding out the plaintext before encryption can be used to increase the Entropy of the ciphertext.

Other methods used for deniable encryption involve establishing a number of abstract 'layers' that are decrypted to yield different plaintexts for different keys. Some of these layers are designed to include so-called 'chaff layers'. These are layers that are composed of random data which provide the owner of the data to have plausible deniability of the existence of layers containing the real ciphertext data. The user can store 'decoy files' on one or more layers while denying the existence of others, identifying the existence of chaff layers as required. The layers are based on file systems that are typically stored in a single directory consisting of files with filenames that are either randomized (in the case where they belong to chaff layers), or are based on strings that identify cryptographic data, the timestamps of all files being randomized throughout.

[1]Illustrating the potential of paratroopers and so initiating the Allied development of their own airborne divisions.

E. Steganographic Encryption and Disinformation

It is arguable that disinformation should, where possible, be used in conjunction with the exchange of encrypted information which has been camouflaged using steganographic techniques for hiding the ciphertext. For example, suppose that it had been assumed by Germany that the Enigma ciphers were being compromised by the British during the Second World War. Clearly, it would have been strategically advantageous for Germany to propagate disinformation using Enigma. If, in addition, 'real information' had been encrypted differently and the ciphertexts camouflaged using broadcasts through the German home radio service, for example, then the outcome of the war could have been very different. The use of new encryption methods coupled with camouflage and disinformation, all of which are dynamic processes, provides a model that, while not always of practical value, is strategically comprehensive and has only rarely been fully realised.

III. Cryptanalysis and the Management of Encryption Technology

There are numerous historical examples where cryptanalysis has been used to decrypt encrypted messages. Any cryptographic system must be able to withstand cryptanalysis [12]. Cryptanalysis methods depend critically on the encryption techniques which have been developed and are therefore subject to delays in publication. Cryptanalysts work on 'attacks' to try and break a cryptosystem. In many cases, the cryptanalysts are aware of the algorithm used and will attempt to break the algorithm in order to compromise the keys or gain access to the actual plaintext. It is worth noting that even though a number of algorithms are freely published, this does not mean that they are the most secure.

Most government institutions and the military do not reveal the type of algorithm used in the design of a cryptosystem. The rationale for this is that, if it is difficult to break a code with knowledge of the algorithm then how difficult is it to break a code if the algorithm is unknown? On the other hand, within the academic community, security in terms of algorithm secrecy is not considered to be of high merit and publication of the algorithm(s) is always recommended. It remains to be understood whether this is a misconception within the academic world (due in part to the innocence associated with academic culture) or a covertly induced government policy (by those who are less innocent!).

The 'known algorithm' approach originally comes from the work of Auguste Kerchhoff. Kerchhoff's Principle states that: *A cryptosystem should be secure even if everything about the system, except the key, is public knowledge*. This principle was reformulated by Claude Shannon as *the enemy knows the system* and is widely embraced by cryptographers world wide. In accordance with the Kerchhoff-Shannon principle, the majority of civilian cryptosystems make use of publicly

known algorithms. The principle is that of 'security through transparency' in which open-source software is considered to be inherently more secure than closed source software. On this basis there are several methods by which a system can be attacked where, in each case, it is assumed that the cryptanalyst has full knowledge of the algorithm(s) used.

A. Example Attack Strategies

Some of principal attack strategies associated with cryptanalysis are as follows:

1) Ciphertext-only Attack: Cipher-only attacks are where the cryptanalyst has a ciphertext of several messages at their disposal. All messages are assumed to have been encrypted using the same algorithm. The challenge for the cryptanalyst is to try and recover the plaintext from these messages. Clearly a cryptanalyst will be in a valuable position if they can recover the actual keys used for encryption.

2) Known-plaintext Attack: Known-plaintext Attack makes the task of the cryptanalysis simpler because, in this case, access is available to both the plaintext and the corresponding ciphertext. It is then necessary to deduce the key used for encrypting the messages, or design an algorithm to decrypt any new messages encrypted with the same key.

3) Chosen-plaintext Attack: Chosen plaitext attacks assume that the cryptanalyst possesses both the plaintext and the ciphertext. In addition, the analyst has the ability to encrypt plaintext and see the ciphertext produced. This provides a powerful tool from which the keys can be deduced.

4) Adaptive-chosen-plaintext Attack: Adaptive-chosen-plaintext Attack is an improved version of the chosen-plaintext attack. In this version, the cryptanalyst has the ability to modify the results based on the previous encryption. This version allows the cryptanalyst to choose a smaller block for encryption.

5) Chosen-ciphertext Attack: Chosen-ciphertext Attack can be applied when the cryptanalyst has access to several decrypted texts. In addition, the cryptanalyst is able to use the text and pass it though a 'black box' for an attempted decrypt. The cryptanalyst has to guess the keys in order to use this method which is performed on an iterative basis (for different keys), until a decrypt is obtained.

6) Chosen-key Attack: Based on some knowledge on the relationship between different keys and is not of practical significance except in special circumstances.

7) Rubber-hose Cryptanalysis: The so called rubber-hose cryptanalysis is based on the use of human factors such as blackmail and physical threat for example. It is often a very powerful attack and sometimes very effective.

8) Differential Cryptanalysis: Differential cryptanalysis is a more general form of cryptanalysis. It is the study of how differences in an input can affect differences in the output. This method of attack is usually based on a chosen plaintext, meaning that the attacker must be able to obtain encrypted ciphertexts for some set of plaintexts of their own choosing. This typically involves acquiring a Crib of some type as discussed in the following section.

9) Linear Cryptanalysis: A known plaintext attack which uses linear relations between inputs and outputs of an encryption algorithm which holds with a certain probability. This approximation can be used to assign probabilities to the possible keys and locate the one that is most probable.

B. Cribs

The problem with any form of plaintext attack is, of course, how to obtain part or all of the plaintext in the first place. One method that can be used is to obtain a Crib. A Crib, a term that originated at Bletchley Park during the Second World War, is a plaintext which is known or suspected of being part of a ciphertext. If it is possible to compare part of the ciphertext that is known to correspond with the plaintext then, with the encryption algorithm known, one can attempt to identify which key has been used to generate the cipherext as a whole and thus decrypt an entire message. But how is it possible to obtain any plaintext on the assumption that all plaintexts are encrypted in their entirety? One way is to analyse whether or not there is any bad practice being undertaken by the user, e.g. sending stereotyped (encrypted) messages. Analysing any repetitive features that can be expected is another way of obtaining a Crib. For example, suppose that a user was writing letters using Microsoft word, for example, having established an electronic letter template with his/her name, address, company reference number etc. Suppose we assume that each time a new letter is written, the entire document is encrypted using a known algorithm. If it is possible to obtain the letter template then a Crib has been found. Assuming that the user is not prepared to share the electronic template (which would be a strange thing to ask for), a simple way of obtaining the Crib could be to write to the user in hardcopy and ask that the response from the same user is of the same type, pleading ignorance of all forms of ICT or some other excuse. This is typical of methods that are designed to seed a response that includes a useful Crib. Further, there are a number of passive Cribs with regard to letter writing that can be assumed, the use of *Dear* and *Yours sincerely*, for example.

During the Second World War, when passive Cribs such as daily weather reports became rare through improvements in the protocols associated with the use of Enigma and/or operators who took their work seriously, Bletchley Park would ask the Royal Air Force to create some 'trouble' that was of little military value. This included seeding a particular area in the North Sea with mines, dropping some bombs on the French coast or, for a more rapid response, asking fighter pilots to go over to France and 'shoot up' targets of opportunity, processes that came to be known as 'gardening'. The Enigma encrypted ciphertexts that were used to report the 'trouble' could then be assumed to contain information such as the name of the area where the mines had been dropped and/or the harbour(s) threatened by the mines. It is worth noting that the ability to obtain Cribs by gardening was made relatively easy because of the war in which 'trouble' was to be expected and to be reported. Coupled with the efficiency of the German war machine with regard to its emphasis on accurate and timely reports, the British were in a privileged position in which they could create Cribs at will and have some fun doing it.

When a captured and interrogated German stated that Enigma operators had been instructed to encode numbers by spelling them out, Alan Turing reviewed decrypted messages, and determined that the number 'eins' appeared in 90% of the messages. He automated the Crib process, creating an 'Eins Catalogue', which assumed that 'eins' was encoded at all positions in the plaintext. The catalogue included every possible position of the various rotors and plug board settings of the Enigma. This provided a very simple and effective way of recovering the key and is a good example of how the statistics (of a word or phrase) can be used in cryptanalysis.

The use of Enigma by the German naval forces (in particular, the U-boat fleet) was, compared to the German army and air force, made secure by using a password from one day to the next. This was based on a code book provided to the operator prior to departure from base. No transmission of the daily passwords was required, passive Cribs were rare and seeding activities were difficult to arrange. Thus, if not for a lucky break, in which one of these code books (which were printed in ink that disappeared if they were dropped in seawater) was recovered in tact by a British destroyer (HMS Bulldog) from a damaged U-boat (U-110) on May 9, 1941, breaking the Enigma naval transmissions under their time-variant code-book protocol would have been very difficult. A British Naval message dated May 10, 1941 reads: '1. Capture of U Boat 110 is to be referred to as operation Primrose; 2. Operation Primrose is to be treated with greatest secrecy and as few people allowed to know as possible...' Clearly, and for obvious reasons, the British were anxious to make sure that the Germans did not find out that U-110 and its codebooks had been captured and all the sailors who took part in the operation were sworn to secrecy. On HMS Bulldog's arrival back in Britain a representative from Bletchley Park, photographed every page of every book. The 'interesting piece of equipment' turned out to be an Enigma machine, and the books contained the Enigma codes being used by the German navy.

The U-boat losses that increased significantly through the decryption of U-boat Enigma ciphers led Admiral Carl Doenitz to suspect that his communications protocol had been compromised. He had no firm evidence, just a 'gut feeling' that something was wrong. His mistake was not to do anything about it, an attitude that was typical of the German High Command who were certifiable with regard to their confidence in the Enigma system. However, they were not uniquely certifiable. For example, on April 18, 1943, Admiral Yamamoto (the victor of Pearl Harbour) was killed when his plane was shot down while he was attempting to visit his forces in the Solomon Islands. Notification of his visit from Rabaul to the Solomon's was broadcast as Morse coded ciphertext over the radio, information that was being routinely decrypted by the Americans. At this point in the Pacific War, the Japanese were using a code book protocol similar to that used by the German Navy, in which the keys were changed on a daily basis, keys that the Americans had 'generated' copies of. Some weeks before his visit, Yamamoto had been given the option of ordering a new set of code books to be issued. He had refused to give the order on the grounds that the logistics associated with transferring new code books over Japanese held territory was incompatible with the time scale of his visit and the possible breach of security that could arise through a new code book being delivered into the hands of the enemy. This decision cost him his life. However, it is a decision that reflects the problems associated with the distribution of keys for symmetric cryptosystems especially when a multi-user protocol needs to be established for execution over a wide communications area. In light of this problem, Yamamoto's decision was entirely rational but, nevertheless, a decision based on the assumption that the cryptosystem had not already been compromised.

The principles associated with cryptanalysis that have been briefly introduced here illustrate the importance of using a dynamic approach to cryptology. Any feature of a security infrastructure that has any degree of consistency is vulnerable to attack. This can include plaintexts that have routine phrases such as those used in letters, the key(s) used to encrypt the plaintext and the algorithm(s) used for encryption.

C. Management of Encryption Technologies

If there can be such a thing as a 'golden rule' associated with the management of encryption technologies, then it is: *never underestimate the enemy*! This includes appreciating that a clever invention or tactic or both could have been anticipated by the opponent or have been developed and implemented *a priori*. Good security management processes are based on products, procedures and protocols that are dynamic at least for the management of systems designed to secure data. However, for the 'code breakers', dynamic systems are notoriously difficult to attack and information security is therefore best undertaken by changing the characteristics of the 'game'. This approach needs to be realized within the context that 'gaming' is valid at many scales, from the socio-political arena to the level of technical innovation and theoretical detail.

Some of the most innovative ideas in information security with regard to changing the 'game' are the simplest. For example, after the Second World War, as the cold war developed, although the USSR did not have complete knowledge of the code breaking activities of the other allies (the British and Americans, in particular), it was well known that the code breaking activities had been based on information retrieved by intercepting wireless-based communications (i.e. wireless encrypted Morse code). For the USSR the answer was simple. They used land lines until they felt confident of their own wireless-based cryptosystems. This was not an elegant solution but it was simple and highly effective and kept NATO 'in the dark' for decades to come with many attempts being made by the western powers to physically intercept or tap the USSR 'land-lines' with little if any success.

It is an irony that as society becomes more and more 'open and transparent' with regard to information in general, including information on how to construct and break industry standard encryption technology [13], so it becomes increasingly difficult to implement secure policies for the protection of that society. This is due, not only to the increased technical awareness, knowledge and abilities of the younger generations,

but is also a product of the social engineering implemented after the Second World War where deference to authority has slowly but surely ebbed away. This has led to radical changes in social- and geo-politics world wide.

The public development of information security technology is one of the most interesting challenges for state control over the 'information society'. As more and more members of the younger generation become increasingly IT literate, it is inevitable that a larger body of perfectly able minds will become aware of the fact that cryptology is not as difficult as they have been led to believe. As with information itself, the days when cryptology was in the hands of a select few with impressive academic credentials and/or luxury civil service careers are over and cryptosystems can now be developed by those with a diverse portfolio of backgrounds which does not necessarily include a University education. This is reflected in the fact that after the Cold War, the UK Ministry of Defense, for example, developed a strategy for developing products driven by commercially available systems. This Commercial-Off-The-Shelf or COTS approach to defence technology has led directly to the downsizing of the UK scientific Civil Service which, during the Cold War, was a major source of scientific and technical innovation.

The average graduate of today can rapidly develop the ability to write an encryption system which, although relatively simple, possibly trivial and ill-informed, can, by the very nature of its non-compliance to international standards, provide surprisingly good security. This can lead to problems with the control and management of information when increasingly more individuals, groups, companies, agencies and nation states decide that they can 'go it alone' and do it themselves. While each home grown encryption system may be relatively weak, compared to those that have had expert development over many years, have been well financed and been tested against the very best of attack strategies, the proliferation of such systems is itself a source of significant difficulty for any authority whose role is to monitor communications traffic in a way that is timely and cost effective. This is why governments world-wide are constantly attempting to control the use and exploitation of new encryption methods in the commercial sector, e.g. the introduction of legislation concerning the decryption of massages by a company client through enforcement of the Regulation of Investigatory Powers (RIP) Act, UK, 2000 [14]. It also explains the importance of international encryption standards in terms of both public perception and free market exploitation. Government and other controlling authorities like to preside over a situation in which everybody else is confidently reliant for their information security on products that have been developed by the very authorities that encourage their use. The reason for this is based on the fact that modern encryption systems can not be successfully attacked unless serious errors are made by the user (induced or otherwise). In other words, the race between the code makers and the code breakers was won by the former many years ago. Thus, only by managing the users of systems and the systems themselves, can encrypted information be read. The origins of this approach are discussed in the following section.

D. *Perfidious Albion*

A historically important example of *Perfidious Albion* relates to a British approach for managing the users of encryption systems and the systems themselves after the end of the Second World War in the late 1940s and 1950s[2], an approach that has come to dominate the management of encryption systems in most sectors. As the British army advanced into west Germany, many thousands of Enigma machines were captured and stock-piled. An issue arose as to what to do with them. Two options were available: (i) to destroy them; (ii) to make good use of them. The latter decision was taken. Coupled with a 'spin' on the quality of German war technology, many Enigma machines were sold on to governments world wide in order for the British to gather intelligence based at the new Government Central Headquarters in Cheltenham. However, the 'key' to this deception was to maintain a critical silence on the work of Cheltenham. This is why the activities of 'Station X' based at Bletchley Park, England, had to remain secrete for such a significant length of time - nearly thirty years after 'Stationed X' was dismantled on Churchill's orders.

Although encryption technologies have improved radically since the development of Enigma, the management of these technologies has not. The broad strategy is to encourage the use of an encryption standard (including key management) that is known and preferably designed by the very authorities whose principal job is to gather intelligence associated with communications traffic that has been encrypted using the same encryption standard. While this management strategy might be described as an example of 'Perfidious Albion', especially by those who have been managed by it, the strategy has become a central theme in the management of encryption systems world wide. It is a strategy that can only be fully compromised through the development of novel encryption methods that do not conform to a 'standard'.

IV. Stochastic Confusion and Diffusion

In terms of plaintexts, diffusion is concerned with the issue that, at least on a statistical basis, similar plaintexts should result in completely different ciphertexts even when encrypted with the same key. This requires that any element of the input block influences every element of the output block in an irregular fashion. In terms of a key, diffusion ensures that similar keys result in completely different ciphertexts even when used for encrypting the same block of plaintext. This requires that any element of the input should influence every element of the output in an irregular way. This property must also be valid for the decryption process because otherwise an intruder may be able to recover parts of the input from an observed output by a partly correct guess of the key used for encryption. The diffusion process is a function of the sensitivity to initial conditions that all cryptographic systems must have. Further, all cryptographic systems should exhibit an inherent topological transitivity causing the plaintext to be mixed through the action of the encryption process.

[2]Based on comments made by Dr S Singh - http://www.simonsingh.net/ - at a Keynote Address, 'The Science of Secrecy', 20th Irish Signals and Systems Conference, University College Dublin, June 10th - 11th, 2009.

The process of 'confusion' ensures that the (statistical) properties of plaintext blocks are not reflected in the corresponding ciphertext blocks. Every ciphertext must have a random appearance to any observer and be quantifiable through appropriate statistical tests. Diffusion and confusion are processes that are of fundamental importance in the design and analysis of cryptological systems, not only for the encryption of plaintexts but for data transformation in general.

A. Stochastic Confusion

The simplest approach to encrypting plaintext described by the vector $\mathbf{p}$ (e.g. consisting of 7-bit ASCII decimal integers) is to add a cipher denoted by $\mathbf{n}$ which is taken to be a stochastic function consisting of random numbers, i.e. $\mathbf{n}$ is a vector consisting of noise. The ciphertext $\mathbf{c}$ is then given by

$$\mathbf{c} = \mathbf{p} + \mathbf{n} \tag{1}$$

This is an example of a substitution cipher and relies on the use of ciphers that can be regenerated by a 'key' which is the initial condition used to 'drive' a random number generated whose algorithm is known. For a 7-bit ASCII code, if the value of any element of the array $\mathbf{c}$ exceeds 127 then the result is wrapped (e.g. 121+10=3), the output being taken to be the modulo of 127. In practice, this process is usually carried out in 'binary space' by first converting the arrays $\mathbf{p}$ and $\mathbf{c}$ to binary form $\mathbf{p}_b$ and $\mathbf{c}_b$, respectively. The binary ciphertext $\mathbf{c}_b$ is then computed using the XOR operation, i.e.

$$\mathbf{c}_b = \mathbf{n}_b \oplus \mathbf{p}_b$$

The plaintext is given by

$$\mathbf{p}_b = \mathbf{n}_b \oplus \mathbf{c}_b$$

Irrespective of whether this simplest of encryption schemes is implemented in decimal integer or binary space, it is imperative that the cipher exhibits statistical properties such that there is no bias associated with the frequency of occurrence of any element. In other words the histrogam of $\mathbf{n}$ must be uniformly distributed in order to counteract a statistical attack based on analysing the frequency of occurrence of elements of $\mathbf{p}$ in which the space between each word is the most common. Equivalently, in binary space, it is important that there is no bias towards a 0 or 1 so that the number of bits should be the same in any binary cipher. The additive process associated with equation (1) represents a process of *confusion* based on, what is in effect, the addition of uniformly distributed noise. It is an example of stochastic confusion upon which the majority of both substitution and transposition (or both) ciphers are based. The aim is to generate a maximum entropy cipher in such a way that there is maximum possible diffusion in terms of key dependency (i.e. that a change in any single bit of the key can effect any, and potentially, all bits of the cipher). This is the usual concept in which the term *diffusion* is used in cryptology, the aim being to maximize (in terms of the entropy of the ciphertext) the process of both diffusion and confusion.

B. Stochastic Diffusion

In this paper we consider the diffusion of plaintext in terms of the convolution of a plaintext array with the cipher, the ciphertext being given by

$$c[i] = n[i] \otimes p[i] = \sum_j n[j-i]p[j]$$

where $\otimes$ denotes the convolution sum as defined above.

It is well known that for $x \in (-\infty, \infty)$, the solution to the diffusion equation

$$(D\partial_x^2 - \partial_t)c(x,t) = 0, \quad c(x,0) = p(x)$$

where D is the *Diffusivity* and $p(x)$ is the initial condition, is given by [16]

$$c(x,t) = g(x,t) \otimes_x p(x) = \int_{-\infty}^{\infty} g(y-x,t)p(y)dy$$

where $\otimes_x$ now represents the convolution integral (over independent variable x) as defined above and

$$g(x,t) = \sqrt{\frac{1}{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

Here, the diffusion of $p(x)$ is determined by a convolution with a Gaussian function whose standard deviation is determined by the product of the Diffusivity D and the time t over which the diffusion process occurs. Stochastic diffusion is based on replacing the convolution kernel (i.e. the function g) for a stochastic function n so that

$$c(x,t) = n(x,t) \otimes_x p(x)$$

For any time t, the ciphertext is given by the convolution of the plaintext with the cipher. In this sense, any fixed value of time $t_i, i = 1, 2, 3...$ can be taken to represent the initial value used to generate $n(x)$, i.e. the key used to initiate the random number generating algorithm.

The inverse problem associated with stochastic diffusion as defined above is not as simple as applying an XOR operation to the ciphertext in binary space. In this case we are required to apply a suitable deconvolution process, i.e. to solve the problem: Given c and n, compute p. We can deconvolve by using the convolution theorem giving

$$p(x) = \mathcal{F}_1^{-1}\left[\frac{C(k)N^*(k)}{\mid N(k) \mid^2}\right]$$

where N is the Fourier transform of n, C is the Fourier transform of c, k is the spatial frequency and $\mathcal{F}_1^{-1}$ denotes the (one-dimensional) inverse Fourier transform, i.e. for a piecewise continuous function $f(x)$ with spectrum denoted by $F(k)$,

$$\mathcal{F}_1^{-1}[F(k)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(k)\exp(ikx)dx = f(x)$$

and

$$F(k) = \int_{-\infty}^{\infty} f(x)\exp(-ikx)dx$$

This approach requires regularization in order to eliminate any singularities when $\mid N \mid^2 \rightarrow 0$ through application of a constrained deconvolution filter [17], [18]. It is not appropriate for encryption because the inverse process is fundamentally ill-conditioned. Instead we consider a method which allows the inverse problem to be solved directly by correlation. To do this the cipher needs to be 'pre-conditioned'. We let

$$m(x) = \mathcal{F}_1^{-1}[M(k)]$$

where $\forall k$

$$M(k) = \begin{cases} \frac{N^*(k)}{|N(k)|^2}, & \mid N(k) \mid \neq 0; \\ N^*(k), & \mid N(k) \mid = 0. \end{cases}$$

The ciphertext is then given by

$$c(x) = m(x) \otimes_x p(x)$$

This result allows us to solve the inverse problem by correlating (denoted by $\odot_x$) c with n. This is based on the following analysis: Using the convolution theorem

$$m(x) \otimes_x p(x) \leftrightarrow M(k)P(k)$$

and, using the correlation theorem,

$$n(x) \odot_x c(x) \leftrightarrow N^*(k)C(k)$$

where $\leftrightarrow$ denotes transformation into Fourier space. Thus

$$N^*(k)C(k) = N^*(k)M(k)P(k) = N^*(k)\frac{N^*(k)}{\mid N(k) \mid^2} = P(k)$$

so that

$$p(x) = n(x) \odot_x c(x)$$

The pre-conditioning of a cipher so that decryption can be undertaken using correlation provides a simple solution for utilizing the process of stochastic diffusion to encrypt data. In this paper, the process is applied in 'image space' to watermark a digital image.

V. Digital Image Watermarking

In 'image space', we consider the plaintext to be an image $p(x, y)$ of compact support $x \in [-X, X]$; $y \in [-Y, Y]$. Stochastic diffusion is then based on the following results:

Encryption

$$c(x, y) = m(x, y) \otimes_x \otimes_y p(x, y)$$

where

$$m(x, y) = \mathcal{F}_2^{-1}\left[M(k_x, k_y)\right]$$

and $\forall k_x, k_y$

$$M(k_x, k_y) = \begin{cases} \frac{N^*(k_x,k_y)}{|N(k_x,k_y)|^2}, & \mid N(k_x, k_y) \mid \neq 0; \\ N^*(k_x, k_y), & \mid N(k_x, k_y) \mid = 0. \end{cases}$$

Decryption

$$p(x, y) = n(x, y) \odot_x \odot_y c(x, y)$$

Here, k_x and k_y are the spatial frequencies and $\mathcal{F}_2^{-1}$ denotes the two-dimensional inverse Fourier transform. For digital image watermarking, we consider a discrete array $p_{ij}, i = 1, 2, ..., I; j = 1, 2, ..., J$ of size $I \times J$ and discrete versions of the operators involved, i.e. application of a discrete Fourier transform and discrete convolution and correlation sums.

For a host image denoted by $h(x, y)$, we consider a watermarking method based on the equation

$$c(x, y) = Rm(x, y) \otimes_x \otimes_y p(x, y) + h(x, y)$$

where

$$\|m(x, y) \otimes_x \otimes_y p(x, y)\|_\infty = 1$$

and

$$\|h(x, y)\|_\infty = 1$$

By normalising the terms in this way, the coefficient $0 \leq R \leq 1$ can be used to adjust the relative magnitudes of the terms such that the diffused image $m(x, y) \otimes_x \otimes_y p(x, y)$ becomes a perturbation of the 'host image' (covertext) $h(x, y)$. This provides us with a way of digital watermarking [19] one image with another, R being referred to as the 'watermarking ratio', a term that is equivalent, in this application, to the standard term 'Signal-to-Noise' or SNR as used in signal and image analysis. For colour images, the method can be applied by decomposing the image into its constituent Red, Green and Blue components. Stochastic diffusion is then applied to each component separately and the result combined to produce an colour composite image.

For applications in image watermarking, stochastic diffusion has two principal advantages:

- a stochastic field provides uniform diffusion;
- stochastic fields can be computed using random number generators that depend on a single initial value or seed (i.e. a private key).

A. Binary Image Watermarks

Watermarking a full grey level or colour image with another grey or colour image, respectively, using stochastic diffusion leads to two problems: (i) it can yield a degradation in the quality of the reconstruction especially when R is set to a low value which is required when the host image has regions that are homogeneous; (ii) the host image can be corrupted by the watermark leading to distortions that are visually apparent. Points (i) and (ii) lead to an optimisation problem with regard to the fidelity of the watermark and host images in respect of the value of the watermark ratio that can be applied limiting the type of host images that can be used and the fidelity of the 'decrypts'. However, if we consider the plaintext image $p(x, y)$ to be of binary form, then the output of stochastic diffusion can be binarized to give a binary ciphertext. The rationale for imposing this condition is based on considering a system in which a user is interested in covertly communicating documents such as confidential letters and certificates, for example.

If we consider a plaintext image $p(x, y)$ which is a binary array, then stochastic diffusion using a pre-conditioned cipher $0 \leq m(x, y) \leq 1$ consisting of an array of floating point numbers will generate a floating point output. The Shannon

Information Entropy of any array $A(x_i, y_i)$ with Probability Mass Function (PMF) $p(z_i)$ is given by

$$I = -\sum_{i=1} p(z_i) \log_2 p(z_i)$$

The information entropy of a binary plaintext image (with PMF consisting of two components whose sum is 1) is therefore significantly less than the information entropy of the ciphertext image. In other words, for a binary plaintext and a non-binary cipher, the ciphertext is data redundant. This provides us with the opportunity of binarizing the ciphertext by applying a threshold T, i.e. if $c_b(x, y)$ is the binary ciphertext, then

$$c_b(x, y) = \begin{cases} 1, & c(x, y) > T \\ 0, & c(x, y) \leq T \end{cases} \tag{2}$$

where $0 \leq c(x, y) \leq 1 \forall x, y$. A digital binary ciphertext image $c_b(x_i, y_j)$ where

$$c_b(x_i, y_i) = \begin{cases} 1, & \text{or} \\ 0, & \text{for any } x_i, y_j \end{cases}$$

can then be used to watermark an 8-bit host image $h(x, y), h \in [0, 255]$ by replacing the lowest 1-bit layer with $c_b(x_i, x_j)$. To recover this information, the 1-bit layer is extracted from the image and the result correlated with the digital cipher $n(x_i, y_j)$. Note that the original floating point cipher n is required to recover the plaintext image and that the binary watermark can not therefore be attacked on an exhaustive XOR basis using trial binary ciphers. Thus, binarization of a stochastically diffused data field is entirely irreversible, i.e. equation (2) describes a one-way function.

B. Statistical Analysis

The expected statistical distribution associated with stochastic diffusion is Gaussian. This can be shown if we consider a binary plaintext image $p_b(x, y)$ to be described by a sum of N delta functions where each delta function describes the location of a non-zero bit at coordinates (x_i, y_j). Thus if

$$p_b(x, y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(x - x_i)\delta(y - y_j)$$

then

$$\begin{aligned} c(x, y) &= m(x, y) \otimes_x \otimes_y p(x, y) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} m(x - x_i, y - y_j). \end{aligned}$$

Each function $m(x - x_i, y - y_j)$ is just $m(x, y)$ shifted by x_i, y_j and will thus be identically distributed. Hence, from the Central Limit Theorem

$$\Pr[c(x, y)] = \Pr\left[\sum_{i=1}^{N} \sum_{j=1}^{N} m(x - x_i, y - y_j)\right] =$$

$$\prod_{i=1}^{N}{}_{\otimes} \Pr[m(x, y)] \equiv \Pr[m(x, y)] \otimes_x \otimes_y \Pr[m(x, y)] \otimes_x \otimes_y ...$$

$$\sim \text{Gaussian}(z), \quad N \to \infty$$

where Pr denotes the Probability Density Function. We can thus expect $\Pr[c(x, y)]$ to be normally distributed and for $m(x, y) \in [0, 1] \forall x, y$ the mode of the distribution will be of the order of 0.5. This result provides a value for the threshold T in equation (2) which for $0 \leq c(x, y) \leq 1$ is 0.5 (theoretically). Note that if $n(x, y)$ is uniformly distributed and thereby represents δ-uncorrelated noise then both the complex spectrum N^* and power spectrum $\mid N \mid^2$ will also be δ-uncorrelated and since

$$m(x, y) = \mathcal{F}_2^{-1}\left[\frac{N^*(k_x, k_y)}{\mid N(k_x, k_y) \mid^2}\right]$$

$\Pr[m(x, y)]$ will be uniformly distributed. Also note that the application of a threshold which is given by the mode of the Gaussian distribution, guarantees that there is no statistical bias associated with any bit in the binary output, at least, on a theoretical basis. On a practical basis, this needs to be computed directly by calculating the mode from the histogram of the cipher and that bit equalization can not be guaranteed as it will depend on: (i) the size of the images used; (ii) the number of bins used to compute the histogram.

C. Principal Algorithms

The principal algorithms associated with the application of stochastic diffusion discussed so far are as follows:

Algorithm I: Encryption and Watermarking Algorithm

Step 1: Read the binary plaintext image from a file and compute the size $I \times J$ of the image.

Step 2: Compute a cipher of size $I \times J$ using a private key and pre-condition the result.

Step 3: Convolve the binary planitext image with the pre-conditioned cipher and normalise the output.

Step 4: Binarize the output obtained in Step 3 using a threshold based on computing the mode of the Gaussian distributed ciphertext.

Step 5: Insert the binary output obtained in Step 4 into the lowest 1-bit layer of the host image and write the result to a file.

The following points should be noted:

(i) The host image is taken to be an 8-bit or higher grey level image which should be of the same size as the plaintext image or else resized accordingly. However, in resizing the host image, its proportions should be the same so that the stegotext image does not appear to be a distorted version of the covertext image. For this purpose, a library of host images should be developed whose dimensions are set according to a predetermined application where the size of the expected plaintext images are known.

(ii) Pre-conditioning the cipher and the convolution processes are undertaken using a Discrete Fourier Transform (DFT).

(iii) The output given in Step 3 will include negative floating point numbers upon taking the real component of a complex array. The array must be rectified by adding the largest negative value in the output array to the same array before normalisation.

(iv) For colour host images, the binary ciphertext can be inserted in to one or all of the RGB components. Ths provides the facility for watermarking the host image with three binary ciphertexts (obtained from three separate binary documents, for example) into a full colour image. In each case, a different key can be used.

(v) The binary plaintext image should have homogeneous margins in order to minimise the effects of ringing due to 'edge-effects' when processing the data in the spectral domain.

Algorithm II: Decryption Algorithm

Step 1: Read the watermarked image from a file and extract the lowest 1-bit layer from the image.

Step 2: Regenerate the (non-preconditioned) cipher using the same key used in Algorithm I.

Step 3: Correlate the cipher with the input obtained in Step 1 and normalise the result.

Step 4: Quantize and format the output from Step 3 and write to a file.

The following points should be noted:

(i) The correlation operation should be undertaken using a DFT.

(ii) For colour images, the data is decomposed into each RGB component and each 1-bit layer is extracted and correlated with the appropriate cipher, i.e. the same cipher or three ciphers relating to three private keys respectively.

(iii) The output obtained in Step 3 has a low dynamic range and therefore requires to be quantized into an 8-bit image based on floating point numbers within the range max(array)-min(array).

D. Examples Results

An example of stochastic diffusion is given in Figure 4 which shows the diffusion of a binary plaintext image with a uniformly distributed cipher and binarization. Figure 5 shows the histograms associated with each image given in Figure 4 illustrating a realization of the statistical analysis discussed in Section V(B). In particular, we see that the diffusion process generates a Gaussian distributed ciphertext whose mode defines the threshold used to generate the binary ciphertext. Figure 6 shows the decrypts before and after binarization of the ciphertext. In the latter case, an accurate representation of the plaintext is obtained albeit embedded in noise. In this case, the fidelity of the decrypt can be improved further by applying a threshold to binarize the result after application of a Gaussian lowpass filter to reduce the level of the background noise. Post-processes of this type can be used to enhance the fidelity of the decrypt as required using a range of image processing methods in order to improve the readability of the image. The type of post-processes applied depends on whether or not the decrypt is used for other applications such as optical character recognition. Figure 7 shows the result of

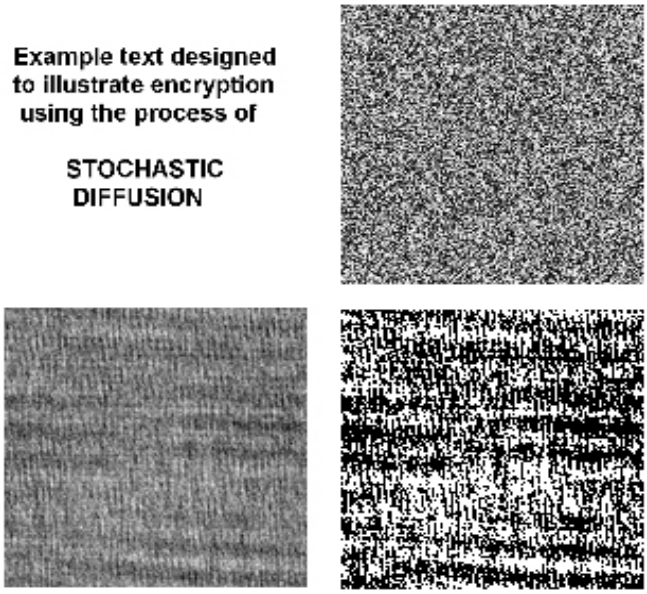

Fig. 4. Illustration of stochastic diffusion using 8-bit grey level images: Binary plaintext image (top-left); uniformly distributed cipher (top-right); grey-level ciphertext after application of stochastic diffusion (bottom-left); binary ciphertext after application of a 50% threshold (bottom-right).

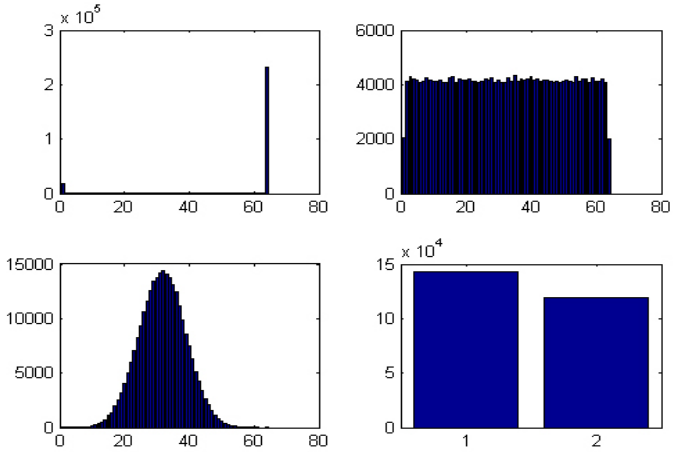

Fig. 5. 64-bin histograms associated with the images given in Figure 4.

applying Algorithm I and Algorithm II for a full colour 24-bit image. This example illustrates the embedding of a binary ciphertext into the lowest 1-bit level of a host image which provides a facility to 'hide' information in an image without impeding the fidelity of the host image. In this example, the binary cipher has been inserted into each RGB component. This improves the fidelity of the decrypt which becomes the result of contributions from all three colour channels rather than one channel in the case of a grey level image.

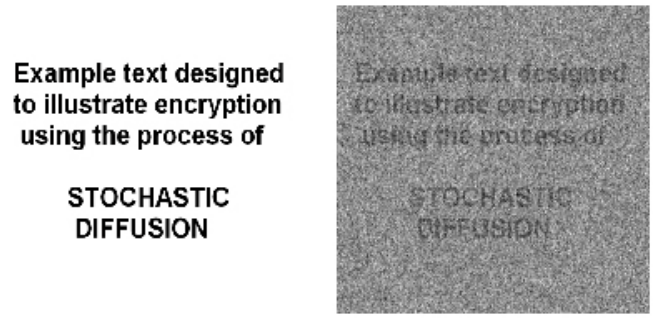

Fig. 6. Decrypts using the grey-level ciphertext (left) and the ciphertext after binarization (right)

Fig. 7. Example of stochastic diffusion applied to a 24-bit colour image (top-left) used to hide an encrypted image of a plaintext image with different font sizes (top-right). The fidelity of the decrypt (bottom-left) is enhanced through the use of a colour host image and watermarking all three colour components with the same watermark. The result can then be post-processed to generate a high quality reconstruction of the original plaintext (bottom-right) which in this case has been generated using a Gaussian lowpass filter with a radius of 1 pixel followed by binarization using a user defined threshold adjusted to give a visually optimal result.

E. 2-bit Randomization

A principal weakness in the algorithms proposed is that the binary watermark is inserted into the lowest 1-bit layer of the host image. The ciphertext can therefore easily be removed and cryptanalysed. A simple solution to this problem is to randomize the watermark over the lowest 2-bit layers of the host image. Thus if $c_b(x_i, x_j)$ is the binary watermark of size $X \times Y$ and $0 \leq r(x_i, y_j) \leq 1$ is an array of uniformly distributed floating point numbers (also of size $X \times Y$) we apply the quantization

$$\forall x_i, y_j \in [X \times Y]$$

$$R(x_i, y_j) = \begin{cases} 0, & 0 \leq r(x_i, y_j) < 0.333; \\ 1, & 0.333 \leq r(x_i, y_j) < 0.666; \\ 2, & 0.666 \leq r(x_i, y_j) < 1; \end{cases}$$

The watermark is then 2-bit randomized by simple array addition to give

$$C(x_i, x_j) = c_b(x_i, x_j) + R(x_i, y_j)$$

The 2-bit array C then replaces the lowest 2-bit layer of the host image. The 1-bit watermark is recovered using array subtraction, i.e.

$$c_b(x_i, x_j) = C(x_i, x_j) - R(x_i, y_j)$$

which requires that r can be reproduced using a cipher generating algorithm with a known 'key' as discussed in Section VII.

VI. StegoCrypt

StegoCrypt is a prototype software system engineered using MATLAB to examine the applications to which stochastic diffusion can be used as compounded in Algorithm I and Algorithm II. It has been designed with a simple Graphical User Interface as shown in Figure 8 whose use is summarised in the following table:

Encryption Mode	**Decryption Mode**
Inputs: Plaintext image Covertext image Private Key (PIN)	*Inputs:* Stegotext image Private key (PIN)
Output: Watermarked Covertext image	*Output:* Decrypted watermark
Operation: Encrypt by clicking on button E (for Encrypt)	*Operation:* Decrypt by clicking on button D (for Dycrypt)

The PIN (Personal Identity Number) can be an numerical

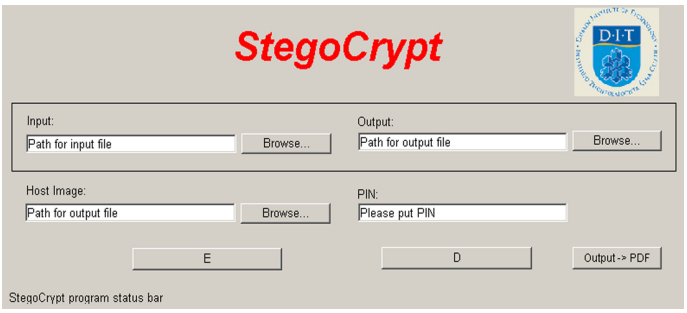

Fig. 8. Graphical User Interface for *StegoCrypt*.

string with upto 16 elements. A demo version of *StegoCrypt* is available from

http://eleceng.dit.ie/arg/downloads/StegoCrypt.zip.

Installation is initiated through setup.exe from the root folder in which the downloaded application has been placed (after unzipping the downloaded file *StegoCrypt.zip*) and following the instructions on screen.

VII. Cipher Generation

The security of a cryptographic system depends on the generation of unpredictable random numbers [24] [25], even though it is difficult to generate a truly random number generator using software-based algorithms [26]. Good random number generators enhance the strength of cryptography and many different methods of generating random numbers have been developed for this purpose. An interesting and relatively simple method is called the *diceware passphrase* [27]. In this method, a list of words is generated and each word numbered. The numbers are generated from an dice, which acts as a random number generator, and are assembled as a five digit number, e.g. 43146. This number is then used to look up a word in a word list. A major advantage of the Diceware approach is that the level of unpredictability in the passphrase can be easily calculated. Each diceware word adds 12.9 bits of information entropy to the passphrase, i.e. log2(65) bits where five words (slightly over 64 bits) are considered a minimum length.

The best random numbers are created by harnessing natural physical processes, such as radioactive decay, which is known to exhibit truly random behaviour [28]. Emissions may be detected in rapid succession or with relatively long delays between emissions, delays that are unpredictable and random. An emission detector cycles through the alphabet at a fixed rate and outputs a letter when an emission is detected. The cycle then continues until the next emission is detected providing another randomly selected letter and so on, a process that generates genuinely random numbers. For example, HotBit [29] random numbers are generated using a radiation source involving beta decay. A user contacts the server, where upon the output can be downloaded over the web. The random numbers provided by HotBits (http://www.fourmilab.ch/hotbits/) are ideal for stream ciphers. However, because they are not generated by some key dependent encryption algorithm, the entire random number stream needs to be exchanged between sender and receiver rather than the key itself. To do this, the random number stream is itself encrypted.

Linear Congruential Generators (LCGs) are based on an iteration of the type

$$x_{i+1} = (a_1 x_i^m + a_2 x_i^{m-1} + a_3 x_i^{m-2} + ...a_{m+1}) \mathrm{mod} P$$

where P is a prime number, typically a large Mersenne prime number $M_n = 2^n - 1$ where n is chosen for primality such that if M_n is a prime number, then so is n. The coefficients a_j are chosen to produce a uniformly distributed random number stream for initial condition $x[1]$. However, cipher generators of this type are vulnerable to attack and cannot be used in cryptography as they are predictable and any LCG can be broken [30].

For cryptographically secure applications there are a range of cipher generating algorithms that can be implemented which is the theme of this section. Any cipher must be capable of ideally generating a uniformly distributed array of integer decimal or floating point numbers which are then normalised to be in the range 0 to 1 inclusively. However, the ideal statistical output of a cipher is sometimes of secondary importance to the characteristics of the algorithm itself in terms of a complexity theoretic approach and the ciphers discussed in this section are examples of complexity theoretic ciphers.

Each cipher is taken to be determined by some function such that

$$x_{i+1} = function(x_i)$$

where x_1 is the 'key' whose statistical 'performance' (i.e. measure of randomness) depends on a single or set of parameters. For the application discussed here, a two dimensional array $n_{ij} \equiv n[i][j], i = 1, 2, ..., I; j = 1, 2, ..., J$ is generated using the following pseudo coded (no standard) algorithm:

```
// Set initial value, e.g. the key
// or Personal Identity Number > 0.
  x[1]=PIN;
// where PIN is a numerical string.

// Compute a two-dimensional array
// y[i][j] of size IxJ composed of
// elements of array x[k].
     k=2;
     for j=1 to J DO:
         for i=1 to I DO:
             x[k]=FUNCTION(x[k-1]);
             y[i][j]=x[k];
     k=k+1;
         END DO
     END DO
// where FUNCTION is taken to output
// x[i]>=0 forall i.


// Normalise the array.
  for j=1 to J DO:
      for i=1 to I DO:
          n[i][j]=y[i][i]/MAX(y[i][j]);
      END DO
  END DO
// where the function MAX computes
// the maximum array value.
```

A cryptographically secure cipher is required to be utilised for Step 2 of Algorithm I prior to pre-conditioning the data. The following functions are examples of 'stream' cipher functions available for this purpose.

A. Additive Generator [23]

$$function(x_i) = (x_{i-a} + x_{i-b} + ... + x_{i-m}) \mathrm{mod} 2^e$$

where $a, b, ..., m$ and e are assigned integers. The algorithm commences by initialising an array x_i with random numbers (obtained from a LCG, for example, for a known key) so that we can consider the initial state of the generator to be $x_1, x_2, x_3, ...$ Additive generators create very long cycles of random numbers.

B. Blum-Blum-Shub Cipher [31]

$$function(x_i) = x_i^2 \mathrm{mod}(pq)$$

where p and q are two large prime numbers (whose product forms the so called Blum integer) that both have a remainder of 3 when divided by 4, i.e. $p = q = 3\mathrm{mod}4$ which is equivalent to

$$p\mathrm{mod}4 = q\mathrm{mod}4 = 3$$

The initial value x_1 that initiates the generator is given by

$$x_1 = x^2 \mathrm{mod}(pq)$$

where x is the key which is a relatively prime of pq. The security associated with this function lies in the difficulty of factoring pq and the nonlinear nature of the iteration which makes it effectively impossible to predict the output of the generator unless the exact value of x_0 - the 'key' - is known. This is a cyptographically secure pseudorandom bit generator. It is not possible to design a polynomial-time algorithm that, on input of the first k bits of an output sequence, can predict the $(k+1)^{\mathrm{th}}$ bit with a probability significantly greater than 0.5. In other words, given the first k bits of the sequence, there is not a practical algorithm that can even allow us to state that the next bit will be 1 (or 0) with a probability greater than 0.5. This function is unpredictable to the left and to the right meaning that for a given sequence, a cryptanalyst cannot predict the next or previous bit in the sequence. This cipher has a non-iterative form given by

$$x_i = x_0^{2^i \mathrm{mod}[(p-1)(q-1)]} \mathrm{mod}(pq)$$

and is one of the simplest and most efficient complexity-theoretic cipher generators of it type.

C. Blum-Micali Cipher [32]

$$function(x_i) = p^{x_i} \mathrm{mod}(q)$$

where p is a prime and q is an odd prime. This function gets its security from the difficulty of computing discrete logarithms. In order for this generator to be secure, the prime number p needs to be large enough so that computing discrete logarithms $\mathrm{mod}p$ is infeasible.

D. RSA Cipher [33]

$$function(x_i) = x_i^e \mathrm{mod}(pq)$$

where p and q are two large primes and e is an integer which is relatively prime to $(p-1)(q-1)$ and the key x_1 is less than pq.

E. Matthews Cipher [34]

$$function(x_i) = (1+r)\left(1+\frac{1}{r}\right)^r x_i(1-x_i)^r,$$
$$\forall \;\; r \in (0,4]$$

The Matthews cipher is a chaotic cipher based on a modification of the logistic map $x \leftarrow rx(1-x), r \in (1,4]$. The effect of this generalization is to produce chaotic behaviour over a greater 'parameter space' r. Compared to the logistic map which yields full chaos at $r = 4$, the chaotic behaviour of the Matthews map is more extensive providing full chaos for the majority (but not all) values of r between approximately 0.5 and 4. The key is the value of x_1 (the initial condition of type float or double depending on the precision that is used - ideally double precision - such that $0 < x_1 < 1$). In addition, because there is a wide range of chaotic behaviour for the Matthews map, the value of r itself can be used as a primary (or secondary) key.

F. Chaotic Ciphers [36]

One of the problems with chaotic ciphers is that they tend to have relatively low cycle lengths and the statistical distribution of the output is not normally uniform. Thus, in addition to having to use floating point arithmetic (usually with double precision), statistical partitioning methods need to be used to output a cipher that is uniformly distributed. Chaotic ciphers are therefore computationally inefficient. The principle advantages of using chaotic ciphers are: (i) their non-reliance on the use of prime numbers which place conditions on the characteristics and arithmetic associated with an encryption algorithm (e.g. the Blum-Blum-Shub and RSA algorithms); (ii) the potentially unlimited number of chaos based algorithms that can be 'invented' to produce a meta-encryption engine, i.e. a cipher based on the application of multiple functions for encrypting data on a block-by-block basis. In this case, a LCG, or other cipher (or even a 'master' chaotic cipher) is used to generate random block lengths L_j to which randomly selected functions $function_{R_j}$ are applied:

$$function[x(i)] = \begin{cases} function_{R_1}[x(i)], & i \in [1, L_1] \\ function_{R_2}[x(i)], & i \in [L_1+1, L_2] \\ \vdots & \vdots \\ function_{R_N}[x(i)], & i \in [L_{N-1}+1, L_N] \end{cases}$$

where L_N is the length of the cipher and R_j is a random integer that 'points' to a function in a database containing a library of upto N functions. This multi-algorithmic approach is an extension of the 'M Algorithm' [37] which is a method for combining multiple pseudo random streams that increases their security where one generators output is used to select a delayed output from another generator. The ciphers required for the system discussed in Section VI do not require a long cycle length unless the image is very large and the downloadable demonstration version of *StegoCrypt* therefore uses a single chaotic cipher. The seed is generated from the PIN by placing a decimal point before the start of the string, i.e. key = .PIN

G. Application of Encryption Standards

In principal, any existing encryption algorithm or encryption system can be used to generate the cipher required by *StegoCrypt* by encrypting an image composed of random noise. The output is then needs to be converted into a decimal integer array and the result normalised as required, i.e. depending on the format of the output that is produced by a given system. In this way, *StegoCrypt* can be used in conjunction with any existing encryption standard.

VIII. Applications

The principal aim of the approach described in this paper is to encrypt an image and transform the ciphertext into a binary array which is then used to watermark a host image. This provides a general method for hiding encrypted information in 'image-space'. In this sense, we have developed a covert encryption method for Electronic Data Interchange (EDI) and, in this section, we focus on some specific applications of value to EDI.

A. Electronic Document Authentication

Electronic or E-documents consisting of letters and certificates, for example, are routinely used in EDI. EDI refers to the structured transmission of data between organizations by electronic means. It is used to transfer electronic documents from one computer system to another; from one trading partner to another trading partner, for example [21]. The USA National Institute of Standards and Technology defines EDI as *the computer-to-computer interchange of strictly formatted messages that represent documents other than monetary instruments* [22]. EDI remains the data format used by the vast majority of electronic transactions in the world and EDI documents generally contain the same information that would normally be found in a paper document used for the same organizational function.

In terms of day-to-day applications, EDI relates to the use of transferring documents between two parties in terms of an attachment. For hardcopies, the attachment is typically the result of scanning the document and generating an image which is formatted as a JPEG or PDF (Print Device File) file, for example. This file is then sent as an attachment to an email which typically refers to the attachment, i.e. the email acts as a covering memorandum to the information contained in the attachment. However, a more common approach is to print a document directly to PDF file, for example. Thus, letters written in MicroSoft word, for example, can be routinely printed to a PDF file for which there are a variety of systems available, e.g. PDF suite http://pdf-format.com/suite/.

For letters and other documents that contain confidential information, encryption systems are often used to secure the document before it is attached to an email and sent. The method discussed in this paper provides a way of encrypting a document using stochastic diffusion and then hiding the output in an image, thus providing a covert method of transmitting encrypted information. However, the approach can also be used to authenticate a document by using the original document as a 'host image'. In terms of the *StegoCrypt* GUI shown in Figure 8, this involves using the same file for the *Input* and *Host Image*. An example of this is shown in Figure 9 where a hardcopy issue of a certificate has been scanned into electronic form - to a .bmp image file. The properties of the image are as follows: File size=3.31Mb; Pixel Dimensions - Width=884 pixels, Height =1312 pixels; Document Size - Width=39.5 cm, Height=46.28cm; Resolution=28 pixels/cm. The result has been encrypted and binarised using stochastic diffusion and the output used to watermark the original document. The fidelity of the decrypt is perfectly adequate to authenticate aspects of the certificate such as the name and qualifications of the holder, the date and signatures, for example. Figure 10 shows the 'Coat of Arms' and the signatures associated with the decrypt given in Figure 9. These results illustrate that the decrypt is adequately resolved for the authentication of the document as a whole. It also illustrates the ability for the decrypt to retain the colour of the original plaintext image.

Fig. 9. Certificate with binary watermark (left) and decrypt (right).

Fig. 10. 'Coat of Arms' (left) and signatures (right) of decrypt given in Figure 9.

B. Authentication of Electronic Letters

When a document is scanned, noise is generated and becomes an inherent feature of the image even though it may not be visually intrusive, e.g. the image of the scanned certificate given in Figure 9. Low level scan noise in a digital image of a scanned document with a uniform (typically white) background is of value in camouflaging the existence of a

1-bit or randomized 2-bit based watermark. However, when a digital image of a letter is generated directly, no noise is generated. This provides a method of revealing the existence, or otherwise, of the watermark. For example, Figure 11 shows an example letter generated by 'printing' it directly to an image file format from the MS Word Editor in which it is composed. If the original image is histogram equalized [35], the white background of the image is not altered because it is perfectly homogenous. However, if the watermarked image is histogram equalized, the existence of the watermark is revealed which defeats the covertness of the approach.

The solution is to add low level noise to the image before inserting the watermark so that it is not clear as to whether the information revealed through histrogram equalization or through direct statistical inspection of the image is due to the presence of a watermark or background noise. For users of MicroSoft word, the solution is to use a texture (*Format→Background→Fill Effect...*) to generate an inhomogeneous background or to include a *Picture Watermark* (*Format→Background→Printed Watermark...*) with *Washout*. Figure 12 shows an example of applying this method to 'camouflage' a 1-bit watermark. In this case, addition of a texture leads to an indeterminacy as to whether analysis of the lowest 1-bit or 2-bit layer of the image is a watermark, randomized watermark or otherwise.

One of the weaknesses of watermarking a letter with itself (a form of self-authentication) is the range of Cribs that are available to an attacker who has extracted the watermark and undertaking cryptanalysis. Thus, ideally, features such as the letter headings and date, for example, should be eliminated from the host image before encryption is undertaken, leaving just the text and important features such as the signature, for example, as agreed by the user(s).

C. Propagation of Disinformation

The technique provides a method of propagating disinformation. Instead of watermarking a letter with itself in order to authenticate the information the letter contains, one letter can be watermarked with a different letter. The host letter is then designed to provide disinformation on the assumption that the email to which it has been 'attached' is intercepted. The watermark is taken to contain the genuine information which could be of an encrypted (alphanumeric) type. In order to encourage an attack, assessment and possible reaction to the disinformation, the attachment can be encrypted using a standard encryption system to a 'strength' compatible with the expectations the intended attacker and recipient of the disinformation has with regard to the source of the data. In other words, the strength of the encryption used should reflect the security interests of the sender as perceived by the attacker.

D. Plausible Deniability

Another reason for encrypting stegotext is to provide a solution that allows a sender to decrypt the data to provide plausible information if forced to do so through a Rubber-hose Cryptanalysis. With regard to this application, it is possible to develop different encrypted information which is embedded

Fig. 11. Example of a plaintext image of a letter (above) after histogram equalisation with (bottom) and without (centre) inclusion of a binary watermark.

Fig. 12. Example of a plaintext image to which a background texture has been applied before introducing a 1-bit watermark (above) and the associated decrypt (below).

into the host image at additional layers. Thus, in addition to replacing the lowest 1-bit layer of the host image with the binary ciphertext, the next lowest 1-bit layer is replaced with another binary ciphertext. If $c_{b,1}[i][j]$ and $c_{b,2}[i][j]$ are two distinct binary cipheretext images, both consisting of elements that are either 0 or 1, then the first and second 1-bit layers of the covertext image are replaced with $c_{b,1}[i][j]$ and $2+c_{b,2}[i][j]$ respectively. This process can be repeated for further layers depending on the characteristics of the host image, i.e. the redundancy of pixel values in each 1-bit layer with regard to the fidelity of the stegotext image.

E. Key-Exchange

In order to use *StegoCrypt* effectively the system must be designed with: (i) a cryptographically secure cipher generator; (ii) a key exchange algorithm. Point (i) has already been discussed in Section VII. With regard to point (ii) there are a range of key exchange algorithms available that can be implemented [32]. A common solution is to utilise the RSA algorithm, not to encrypt plaintext, but to encrypt and transmit the keys used to drive a symmetric encryption system of which *StegoCrypt* is a typical example.

The RSA algorithm is named after Ronald Rivest, Adi Shamir and Leonard Adelman, Computer Science researchers at the Massachusetts Institute of Technology, who developed and patented the algorithm in 1977. RSA gets its security from the difficulty of factoring large numbers [38]. The public and private keys are functions of a pair of large (100 to 200 digits or even larger) prime numbers. Recovering the plaintext from the public key and the cipher text is conjectured to be equivalent to factoring the product of the two primes. The success of this algorithm, which is one of the oldest and most popular public key cryptosystems, is based on difficulty of factoring. For completeness, A brief overview of the RSA algorithm is given in Appendix II.

IX. Discussion

The use of the internet to transfer documents as image attachments has and continues to grow rapidly as part of a global EDI infrastructure. It is for this 'market' that the approach reported in this paper has been developed. Inserting a binary watermark into a host image obtained by binarizing a floating point citertext of a document provides a cryptographically secure solution. Although the watermark can be easily removed from the covertext image - unless 2-bit randomization is implemented as discussed in Section V(E) - it can not be decrypted without the recipient having access to the correct cryptographically secure algorithm and key. The encrypted watermark is not 'suspicious' especially when a document has background scan noise, for example, or a background texture has been introduced as discussed in Section VIII(B).

The key-exchange approach discussed in Section VIII(E) is typical of an infrastructure based on a single sender-receiver scenario in which the use of a system such as *SegoCrypt* is downloaded and installed by both parties or is accessible on-line. In this case, the algorithm used to generate the cipher is the same and the security is based on the PIN which is exchanged using a Public Key Infrastructure (PKI).

Another approach is to use an open key approach but provide a version of *StegoCrypt* that has a unique cipher generating algorithm designed specifically for the 'service provider'. For example, many institutes such as universities still issue 'paper certificates' to their graduates. These certificates are then scanned and sent as attachments along with a CV and covering letter when applying for a job. It is at this point that the certificate may be counterfeited and, for this reason, some establishments still demand originals to be submitted. *StegoCrypt* provides the facility to issue electronic certificates (in addition or in substitution to a hardcopy) which can then

be authenticated as discussed in Section VIII(A). By including a serial number on each certificate (a Certificate Identity Number) which represents an 'open key', the document can be submitted to the authority that issued the certificate for authentication, for which an online service can be established as required. The security associated with this scenario relies on the 'authority' having and using a unique complexity theoretic cipher generator. This 'Private Algorithm Infrastructure' or PAI is at odds with Kerchhoff-Shannon principle (i.e. that the enemy knows the system) but in light of the issues discussed in Section III(D), provides the opportunity for an authority to introduce their own 'private practice' subject to any regulation of investigatory powers.

APPENDIX I
TRANSPOSITION CIPHERS

A transposition cipher changes the order in which characters arise in a plaintext [39], [40]. A bijective function is used on the characters' positions to encrypt and an inverse function to decrypt. A function f from a set X to a set Y is said to be bijective if for every y in Y there is exactly one x in X such that $y = f(x)$. In other words, f is bijective if there exists a one-to-one correspondence between the sets.

A typical approach to designing a transposition cipher is to use a so called Columnar transposition. Consider the plaintext

```
THE CAT SAT ON THE MAT
```

and the keyword

```
TIGER
```

which is converted into a numerical array by the alphabetical order in which the letters occur (i.e. 53214). We consider a set of columns that depend on the length of the keyword that is used and write the plaintext into the resulting grid; thus

```
53214
THE C
AT SA
T ON
THE M
AT
```

The columns are then written out in the numerical order associated with the keyword, i.e.

```
 SN  E OE HT HTCA M TATTA
```

If a keyword is chosen that has multiple occurrences of a letter, then we can consider two approaches:

- the letters are treated as if they are the next letter in alphabetical order, e.g. the keyword *MAXIMUS* yields a numeric keystring of *3172465*.
- recurrent keyword letters are numbered identically, e.g. the keyword *MAXIMUS* yields the numerical keystring *3162354*. This is the basis for the Myszkowski transposition cipher in which plaintext columns with unique keystring numbers are transcribed down each column and those with recurring numbers are transcribed left to right.

A. Example Transposition Ciphers

1) The Bifid Cipher: Consider the following 5 x 5 Polybius square with a randomised alphabet

```
  1 2 3 4 5
1 B G W K Z
2 Q P N D S
3 I O A X E
4 F C L U M
5 T H Y V R
```

which is the 'key' to the following encryption processes in which we consider the plaintext *TRANSPOSITION*. Coordinate convention is undertaken where the coordinates are written below each letter, thus,

```
T R A N S P O S I T I O N
5 5 3 2 2 2 3 2 3 5 3 3 2
1 5 3 3 5 2 2 5 1 1 1 2 3
```

and then output on a row by row basis:

```
553222323533215335225111 23
```

By dividing this integer stream up into coordinate pairs, the ciphertext can be generated using the same Polybius square key, i.e.

```
55 32 22 32 35 33 21 53 35 22 51 11 23
R  0  P  O  E  A  Q  Y  E  P  T  B  N
```

Note that each ciphertext character depends on two plaintext characters and hence, the Bifid cipher is a *digraphic* cipher. To decrypt, the procedure is simply reversed. With regard to longer messages, block encryption is used. As with all block encryption techniques, the plaintext is first broken up into blocks of either fixed or random length. Each block is then encrypted separately. For random block encryption, an algorithm/key is used that randomises the length of each block.

2) The Trifid Cipher: The Trifid cipher is based on the principles of the Bifid cipher but instead of using a two dimensional grid we consider a three dimensional grid. Three coordinates are assigned to each letter or symbol to form the key. Thus, consider a $3 \times 3 \times 3 = 27$ cube where

```
A 111  F 112  N 113
I 121  U 122  V 123
G 131  K 132  O 133
L 211  D 212  T 213
Q 221  R 222  J 223
B 231  H 232  Y 233
M 311  S 312  X 313
E 321  P 322  W 323
C 331  Z 332
```

The plaintext *transposition* now becomes

```
T R A N S P O S I T I O N
2 2 1 1 3 3 1 3 1 2 1 1 1
1 2 1 1 1 2 3 1 2 1 2 3 1
3 2 1 3 2 2 3 2 1 3 1 3 3
```

giving the number stream

```
221133131211112111231212313213223213133
```

which, converting into triplets gives the ciphertext

```
221 133 131 211 112 111 231
Q   O   G   L   F   A   B

212 313 213 223 213 133
D   X   T   J   T   O
```

The method can of course be extended to higher dimensions. Further each dimension can be used to represent one bit in the binary representation of a symbol. In other words, the principles associated with using *Bifid* and *Trifid* encryption can be applied in binary form using a scrambled 7-bit ASCII table, for example, which forms a 'key' to the encryption/decryption process.

B. Anagramming

A single transposition cipher can be attacked by guessing column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible word associations. To increase its strength, the columnar transposition can be applied twice where the same key is used in each case or two different keys are used - double encryption. There are, as usual, a number of variations that can be applied to this method. Further, a random number generator can be used to produce the numerical array that controls the order in which the columns are used, the keyword being used as the initial condition, for example.

Because transposition does not affect the frequency distribution of a plaintext, statistical analysis is an obvious way for a cryptanalysis to launch an attack. If the ciphertext exhibits a frequency distribution very similar to plaintext, it is most likely the result of transposition or a poorly designed substitution cipher. Upon concluding that a transposition cipher has been applied, the ciphertext can be attacked by anagramming. Anagramming is based on shifting partitioned sections of the ciphertext and looking for sections that look like anagrams of English words and then solving the anagrams. Once such anagrams have been found, they reveal information about the transposition pattern and can consequently be extended. Simple transposition ciphers have the property that if a key is tried that is very close to the correct key, then it can reveal sections of legible plaintext interspersed by scrambled data. Consequently such ciphers are vulnerable to optimum seeking algorithms such as genetic algorithms.

C. Fractionation and Diffusion

The weaknesses associated with both substitution and transposition ciphers used separately can be overcome by combining the two. For example, a ciphertext generated through a substitution cipher can then be transposed. Anagramming the transposition cipher is then invalid because of the substitution process applied *a priori*. This combination approach is particularly powerful if combined with fractionation. Fractionation is one way of generating diffusion in a cipher, which in terms of plaintexts is designed to ensure that, at least on a statistical basis, similar plaintexts do not result in similar ciphertexts even when encrypted using the same key.

Fractionation is a pre-process in which each plaintext symbol is divided into a number of ciphertext symbols. For example, if the plaintext alphabet is written out in a grid, then every letter in the plaintext can be replaced by its grid coordinates. This is the rationale associated with the use of the Polybius square

```
  1 2 3 4 5
1 A B C D E
2 F G H I J
3 K L M N O
4 P Q R S T
5 U V W X Y
```

where the word *CHESS* can be represented by the 'coordinates' number stream 1323154444. Originally conceived by Polybius in Ancient Greece as a method for telegraphy, in cryptography, the square (which can be of any size in order to accommodate alphabets and characters sets of different lengths) can be used to fractionate.

Another method of fractionation is to simply convert the message to a code (such as Morse code or an equivalent), with a symbol for spaces as well as dots and dashes. When fractionated plaintexts are transposed, the components of individual letters become widely separated in the message. This achieves a greater level of diffusion, i.e. the diffusivity increases. Note that the transposition process can be implemented by replacing each component of the plaintext with a binary representation using the ASCII code which is also used to convert the new binary string into the corresponding ASCII characters. This process can be iterated in binary form to further strengthen the ciphertext. The disadvantage of combination ciphers, fractionation and further variations on a theme is that they are usually computationally intensive and error prone compared to simpler ciphers. Such ciphers are of historical interest, when computational methods for generating random and/or chaotic numbers was not possible, and many modern encryption techniques are substitution based in which the computational effort is focused on producing cryptographically secure ciphers - pseudo random or pseudo chaotic number streams.

APPENDIX II
RSA ALGORITHM

The basic generator, as presented Section VII(D), can be used to compute a cipher. However, the real value of the algorithm lies in its use for transforming plaintext P_i to ciphertext C_i directly via the equation

$$C_i = P_i^e \mathrm{mod}(pq), \quad e < pq$$

We then consider the decryption process to be based on the same type of transform, i.e.

$$P_i = C_i^d \mathrm{mod}(pq)$$

The problem is then to find d given e, p and q. The 'key' to solving this problem is to note that if $ed - 1$ is divisible by

$(p-1)(q-1)$, i.e. d is given by the solution of

$$de = \mathrm{mod}[(p-1)(q-1)]$$

then

$$C_i^d \mathrm{mod}(pq) = P_i^{ed} \mathrm{mod}(pq) = P_i \mathrm{mod}(pq)$$

using *Fermat's Little Theorem*, i.e. for any integer a and prime number p

$$a^p = a \mathrm{mod} p$$

Note that this result is strictly dependent on the fact that $ed-1$ is divisible by $(p-1)(q-1)$ making e a relative prime of $(p-1)(q-1)$ so that e and $(p-1)(q-1)$ have no common factors except for 1. This algorithm, is the basis for many public/private or asymmetric encryption methods. Here, the public key is given by the number e and the product pq which are unique to a given recipient and in the public domain (like an individuals telephone number). This public key is then used to encrypt a message transformed into a decimal integer array M_i say using the one-way function

$$C_i = M_i^e \mathrm{mod}(pq).$$

The recipient is then able to decrypt the ciphetext C_i with knowledge of p and q which represents the private key obtained by solving the equation

$$de = \mathrm{mod}[(p-1)(q-1)]$$

for d and then using the result

$$M_i = C_i^d \mathrm{mod}(pq).$$

In this way, the sender and receiver do not have to exchange a key before encryption/decryption can take place and such systems, in effect solve the key exchange problem associated with symmetric ciphers. Note that the prime numbers p and q and the number $e < pq$ must be distributed to users in such a way that they are unique to each user on the condition that d exists! This requires an appropriate infrastructure to be established by a trusted third party whos 'business is to distribute values of e, pq and d to its clients a Public Key Infrastructure (PKI). A PKI is required in order to distribute public keys, i.e. different but appropriate values of e and pq for use in public key cryptography (RSA algorithm). This requires the establishment of appropriate authorities and directory services for the generation, management and certification of public keys.

The principal vulnerability of the RSA algorithm with regard to an attack is that e and pq are known and that p and q must be prime numbers - elements of a large but (assumed) known set. To attack the cipher, d must be found. But it is known that d is the solution of

$$de = \mathrm{mod}[(p-1)(q-1)]$$

which is only solvable if $e < pq$ is a relative prime of $(p-1)(q-1)$. An attack can therefore be launched by searching through prime numbers whose magnitudes are consistent with the product pq (which provides a search domain) until the relative prime condition is established for factors p and q. However, factoring pq to calculate d given e is not trivial.

Using typical computing power, factoring pq given e is relatively intractable. It is possible to attack RSA by guessing the value of $(p-1)(p-1)$ but this attack is no easier than factoring pq which is the most obvious means of attack. Any adversary will have the public key, e, pq and to find the decryption key d, the attacker has to factor pq. It is possible for a cryptanalyst to try every possible d but this brute force approach is less efficient than trying to factor pq. The application of dedicating computing facilities for this purpose can also be applied to compute large values of primes. For this reason, the increase in computing power will not necessarily provide a solution for breaking RSA encrypted data. Further, any conceivable method invented for a cryptanalyst to deduce d may also provide a new way to factor large numbers which can, in turn, be used to develop new RSA based encryption. Thus, all that RSA cryptanalysis has shown is that the attacks discovered to date illustrate the pitfalls to be avoided when implementing RSA and although RSA ciphers can be attacked, the algorithm can still be considered secure when used properly. RSA based products are available commercially from a range of providers, e.g. http://www.rsa.com/.

ACKNOWLEDGMENTS

The authors are grateful for the support of the Science Foundation Ireland and Dublin Institute of Technology.

REFERENCES

[1] S. Katzenbeisser and F. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, 2000.
[2] N. F. Johnson, Z. Duric and S. Jajodia, *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures*, Kluwer Academicf Publishers, 2001.
[3] J. Buchmann, *Introduction to Cryptography*, Springer 2001.
[4] O. Goldreich, *Foundations of Cryptography*, Cambridge University Press, 2001.
[5] M. J. Turner, J. M. Blackledge and P. Andrews, *Fractal Geometry in Digital Imaging*, Academic Press, 1997.
[6] P. R. Marie, *Fractal-Based Models for Internet Traffic and Their Application to Secure Data Transmission*, PhD Thesis, Loughborough University, 2007.
[7] W. R. Harwood, *The Disinformation Cycle: Hoaxes, Delusions, Security Beliefs, and Compulsory Mediocrity*, Xlibris Corporation, 2002.
[8] R. Miniter, *Disinformation*, Regnery Publishing, 2005.
[9] T. Newark and J. F. Borsarello, *Book of Camouflage* Brassey's, 2002.
[10] H. Gerrad and P. D. Antill, *Crete 1941: Germany's Lightning Airborne Assault*, Osprey Publishing, 2005.
[11] http://eprint.iacr.org/1996/002
[12] P. Garrett, *Making, Braking Codes*, Prentice Hall, 2001.
[13] Hacker's Black Book, http://www.hackersbook.com
[14] http://www.opsi.gov.uk/acts/acts2000/ukpga_20000023_en_1
[15] S. Singh, *The Code Book: The Secret History of Codes and Code-breaking*, Doubleday, 1999
[16] G. Evans, J. M. Blackledge and P. Yardley, *Analytical Methods for Partial Differential Equations*, Springer Undergraduate Mathematics Series, Springer-Verlag, 2000.
[17] M. Bertero and B. Boccacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, 1998.
[18] J. M. Blackledge, *Digital Signal Processing*, Second Edition, Horwood Publishing, 2006.
[19] I. J. Cox, M. Miller, and J. Bloom, *Digital Watermarking*, Morgan Kaufmann, 2002.
[20] J. M. Blackledge and K. W. Mahmoud, *Printed Document Authentication using Texture Coding*, ISAST Journal on Electronics and Signal Processing, To be Published, 2009.
[21] http://en.wikipedia.org/wiki/Electronic_Data_Interchange

[22] M. Kantor and J. H. Burrows (1996-04-29). Electronic Data Interchange', National Institute of Standards and Technology, 1996 http://www.itl.nist.gov/fipspubs/fip161-2.htm.
[23] R. S. Wikramaratna, *ACORN - A New Method for Generating Sequences of Uniformly Distributed Pseudo Random Numbers*, Journal of Computational Physics, 83, 16-31, 1989.
[24] E. Klarreich, *Take a Chance: Scientists put Ramdomness to work*, Science News, 166(23), 362, 2004; http://www.sciencenews.org/articles/20041204/bob9.asp.
[25] E. Uner, *Generating Random Numbers*, 2004; http://www.embedded.com/showArticle.jhtml?articleID=20900500.
[26] N. Al-Ismaily, *Dynamic Block Encryption with Self-Authenticating Key Exchange*, PhD Thesis, Loughborough University, 2006
[27] A. G. Reinhold, *Diceware Passphrase*, 1995; http://world.std.com/ reinhold/diceware.html.
[28] California Intitute of Technology, *The Geiger Counter and counting Statistics*, 1997; http://www.kronjaeger.com/hv-old/radio/geiger/caltech/exp2.htm.
[29] J. Walker, *HotBits: Genuine random numbers, generated by radioactive decay*, 1998; http://www.fourmilab.ch/hotbits/.
[30] J. C. Lagarias and J. Reeds, *Unique Extrapolation of Polynomial recurrences*, SIAM Journal on Computing, **7**(2), 342-362, 1988.
[31] L. Blum, M. Blum, and S. Shub, *A Simple Unpredictable Random Number Generator*, SIAM Journal of Computing, 15, 1986 http://locus.siam.org/SICOMP/volume-15/art0215025.html.
[32] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Wiley, 1996, ISBN: 0471117099.
[33] W. Alexi, B. Z. Chor, O. Goldreich and C. P. Schnorr, *RSA and Rabin Functions: Certain Parts are as Hard as the Whole*, SIAM Journal on Computing, **17**(2), 194-209, 1988
[34] R. Matthews, *On the Derivation of a Chaotic Encryption Algorithm*, Cryptologia, 13, 2942, 1989.
[35] J. M. Blackledge, *Digital Image Processing*, Horwood Publishing, 2005, ISBN 1-898563-49-7, http://eleceng.dit.ie/papers/103.pdf.
[36] J. M. Blackledge, *Multi-algorithmic Cryptography using Deterministic Chaos with Applications to Mobile Communications*, ISAST Transactions on Electronics and Signal Processing, **2**(1), 23 - 64, 2008, ISSN 1797-2329.
[37] D. Knuth, *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*, Second Edition, Addison-Wesley, 1981.
[38] N. Hahnfield, *Cryptography Tutorial: RSA*, 2001; http://www.antilles.k12.vi.us/math/cryptotut/rsa1.htm.
[39] J. Buchmann, *Introduction to Cryptography*, Springer 2001.
[40] O. Goldreich, *Foundations of Cryptography*, Cambridge University Press, 2001.

Jonathan Blackledge received a BSc in Physics from Imperial College, London University in 1980, a Diploma of Imperial College in Plasma Physics in 1981 and a PhD in Theoretical Physics from Kings College, London University in 1983. As a Research Fellow of Physics at Kings College (London University) from 1984 to 1988, he specialized in information systems engineering undertaking work primarily for the defence industry. This was followed by academic appointments at the Universities of Cranfield (Senior Lecturer in Applied Mathematics) and De Montfort (Professor in Applied Mathematics and Computing) where he established new post-graduate MSc/PhD programmes and research groups in computer aided engineering and informatics. In 1994, he co-founded Management and Personnel Services Limited where he is currently Executive Director. His work for Microsharp (Director of R & D, 1998-2002) included the development of manufacturing processes now being used for digital information display units. In 2002, he co-founded a group of companies specializing in information security and cryptology for the defence and intelligence communities, actively creating partnerships between industry and academia (e.g. Lexicon Data Limited). He is currently holder of the Stokes Chair in Digital Signal Processing and Information and Communications Technology based at Dublin Institute of Technology and has published over one hundred scientific and engineering research papers and technical reports for industry, six industrial software systems, fifteen patents, ten books and been supervisor to sixty research (PhD) graduates. His current research interests include computational geometry and computer graphics, image analysis, nonlinear dynamical systems modelling and computer network security, working in both an academic and commercial context. He holds Fellowships with England's leading scientific and engineering Institutes and Societies including the Institute of Physics, the Institute of Mathematics and its Applications, the Institution of Electrical Engineers, the Institution of Mechanical Engineers, the British Computer Society, the Royal Statistical Society and the Chartered Management Institute.

Dmitry Dubovitskiy received a BSc and Diploma in Aeronautical Engineering from Saratov Aviation Technical College in 1993, an MSc in Computer Science and Information Technology from Baumann Moscow State Technical University in 1999 and a PhD in Computer Science from De Montfort University in 2005 under the supervision of Professor J M Blackledge. As a project leader in medical imaging at Microsharp Limited from 2002 to 2005, he specialized in information systems engineering, developing image recognition systems for medical applications for real time operational diagnosis. He founded Oxford Recognition Limited in 2005 which specialises in the applications of artificial intelligence for computer vision. He has developed a range of computer vision systems for industry including applications for 3D image visualisation and has been coordinator for the INTAS project in distributed automated systems for acquiring and analysing eye tracking data. His current interests are in information and communicatuions security.