2020

Applications of Artificial Intelligence to Cryptography

Jonathan Blackledge

Napo Mosola

SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 3
ISSN 2054-7390

Applications of Artificial Intelligence to Cryptography

[1,2,3,4,5]**Jonathan Blackledge and** [6,7]**Napo Mosola**
[1]*Stokes Professor, Science Foundation Ireland;*
[2]*Honorary Professor, School of Electrical and Electronic Engineering, Technological University Dublin;*
[3]*Visiting Professor, Faculty of Arts, Science and Technology, University of Wales, Wrexham Glyndwr;*
[4]*Professor Extraordinaire, Department of Computer Science, University of Western Cape;*
[5]*Honorary Professor and* [6]*Research Associate, School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal;* [7]*Lecturer in Computer Science, National University of Lesotho.*
jonathan.blackledge@TUDublin; nn.mosola@nul.ls

ABSTRACT

This paper considers some recent advances in the field of Cryptography using Artificial Intelligence (AI). It specifically considers the applications of Machine Learning (ML) and Evolutionary Computing (EC) to analyze and encrypt data. A short overview is given on Artificial Neural Networks (ANNs) and the principles of Deep Learning using Deep ANNs. In this context, the paper considers: (i) the implementation of EC and ANNs for generating unique and unclonable ciphers; (ii) ML strategies for detecting the genuine randomness (or otherwise) of finite binary strings for applications in Cryptanalysis. The aim of the paper is to provide an overview on how AI can be applied for encrypting data and undertaking cryptanalysis of such data and other data types in order to assess the cryptographic strength of an encryption algorithm, e.g. to detect patterns of intercepted data streams that are signatures of encrypted data. This includes some of the authors' prior contributions to the field which is referenced throughout. Applications are presented which include the authentication of high-value documents such as bank notes with a smartphone. This involves using the antenna of a smartphone to read (in the near field) a flexible radio frequency tag that couples to an integrated circuit with a non-programmable coprocessor. The coprocessor retains ultra-strong encrypted information generated using EC that can be decrypted on-line, thereby validating the authenticity of the document through the Internet of Things with a smartphone. The application of optical authentication methods using a smartphone and optical ciphers is also briefly explored.

Keywords: Artificial Intelligence, Artificial Neural Networks, Evolutionary Computing, Machine Learning, Cryptography, Cryptanalysis, Radio Frequency Identification, Optical Authentication.

1 Introduction

The term Artificial Intelligence (AI) covers a range of methodologies and applications that are designed to enable a computer to undertake tasks that are conventionally the domain of human intelligence. AI is 'the ability of a digital computer or a computer-controlled robot to perform tasks commonly associated with intelligent beings' [1]. The principal test of AI is predicated on the Turing Test developed in 1950, by Alan Turing. This is the test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human [2]. The current applications of AI range from speech recognition,

DOI: 10.14738/tmlai.83.8219
Publication Date: 06th June, 2020
URL: http://dx.doi.org/10.14738/tmlai.83.8219

finance, avionics, navigation, gaming, robotics and medicine, for example. This paper has been composed to give an overview of such applications focusing on cryptography, while referencing more technically oriented papers that are relevant to the material. In particular, the paper focuses on the application of Evolutionary Computing and Artificial Neural Networks for generating unique and unclonable non-linear ciphers using real-world noise sources.

1.1 Machine Learning

An important sub-category of AI is Machine Learning (ML). ML is mostly associated with the problem of pattern recognition. This is where a complex dataset of possibly irregular patterns in a signal or an image, for example, is required to be categorized into common features and/or segments which can then be classified in some pre-determined way. These classifications are typically associated with statistical measures computed from a signal and statistical and/or geometric metrics for an image. If a cluster of such metrics into a specific numerical range is sufficiently different to be correlated with known features in the data, then a decision can be made through application of a threshold in order to design a decision-making criterion. The problem is often how to find an optimum threshold to do this, such that the accuracy of the decision taken is optimal, the optimum threshold value being subject to a confidence interval. By making the threshold adapt to the demarcation of certain input metrics in terms of their known accuracy, quantity and other prior information, the logic of the decision-making process can be made 'softer' in terms of its tolerance to the data. This is the basis for implementing so called 'Fuzzy Logic Systems' which provide the foundations for the development and implementation of Artificial Neural Networks (ANNs). An ANN typically increases the accuracy of the decisions associated with the classification of a pattern than can be obtained through conventional data categorization (based on a logical and/or fuzzy logical quantification). The following section considers the basis for this.

1.2 Artificial Neural Networks

A simple ANN is based on inputting a class of metrics that are taken to be a composite representation of the data, metrics that are computed by processing the data to form a so called 'feature vector'. A weight w (with a specific floating-point value) is then applied to determine the significance of each component of this vector (by multiplying each component of the vector with its weight) and the weighted components added together to produce a single output value. By changing the values of these weights, the ANN provides an output in relation to a decision-making process. This is an example of a data transfer process which transforms multiple inputs into a single output. It can be replicated to produce a number of identical or different outputs by inputting an identical set of weighted feature vectors or changing the components of the vector that are input. This generates a 'single layer' of outputs that can be taken to be inputs to some 'hidden layer' where the weights are computed and changed again. By replicating this process n times, we can develop a multi-layer network composed of n layers. Figure 1 depicts an example of a simple, single hidden-layer ANN. In this example, the hidden layer is composed of 4 nodes. The computations involved in the transfer of information from the hidden layer to the output layer are taken to include a change in the numerical value to all or some of the weights that are applied. This is an example of a 'feed-forward network' where the 'information' flows in one direction only, i.e. forward, from the input nodes, through the hidden nodes and on to the output nodes. There are no cycles or feed-back loops in the network. Thus, if the three input nodes given in Figure 1 have numerical values (x_1, x_2, x_3), say, then the inputs to the 4 nodes of the hidden layer are (from top to bottom in Figure 1) given by the elements of the vector:

$$(w_{11}x_1 + w_{12}x_2 + w_{13}x_3,\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3,\ w_{31}x_1 + w_{32}x_2 + w_{33}x_3,\ w_{41}x_1 + w_{42}x_2 + w_{43}x_3)$$

where w_{ij} , $i = 1,2,3,4; j = 1,2,3$ are the 12 weights required to modify the 3 inputs from the input layer as they are fed into the 4 nodes of the hidden layer (indicated graphically by the arrows given in Figure 1).

From the description above, and, with reference to Figure 1, it is clear that there are four issues that need to be considered in the design of any such network of input-outputs: (i) What should the size of the feature vector be, thereby specifying the number nodes associated with the input layer?; (ii) how many outputs are required in the output Layer?; (iii) how many hidden layers should be used and how many nodes should each layer contain?; (iv) how can we automate the process by which the weights are initialized and/or then adjusted? In regard to point (iv), it is necessary to design an algorithm for automation of the adjustment of the weights subject to knowledge of the expected output(s) in the output layer. This requires training data to be provided so that the weights can be updated by comparing the outputs they provide with 'target' data.

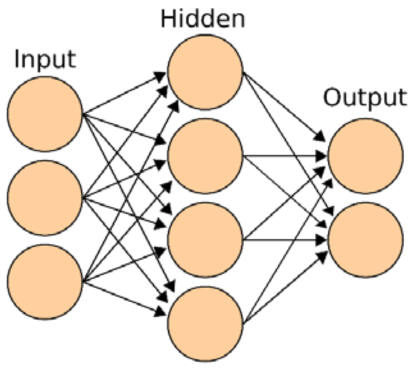


Figure 1. Example of a simple neural network consisting of an input layer with three inputs or nodes (i.e. a three-component feature vector) a single 4-node hidden layer and an output layer consisting of two output nodes [3].

There are a number of different, but closely related methods (algorithms) to train an ANN by adjusting the weights until an output to the network for a given input is the same as the training data (within a specified tolerance). The original and arguably the most common approach, is to apply the back-propagation algorithm [4] for application to feed forward networks. This is based on the Gradient Descent Method; an iterative optimization algorithm for finding a local minimum of a differentiable function. By computing the gradient of a loss function with respect to all the weights (taken to be a discrete vector) and iterating the process, the weights can be adjusted to generate an output that is close to the training data. The algorithm has a synergy to the application of a least squares method for solving a system of linear equations (especially over-determined systems) subject to minimizing an error function and is an example of dynamic programming. Application of the chain rule is used to compute the gradient of the loss function with respect to each weight and a threshold function applied which determines whether or not the value from one node should 'flow' further (i.e. continue on to the next node(s) in the next layer).

There are a wide range of AI software and systems available for the development of AI applications, e.g. [5] and [6], respectively. MathWorks MATLAB, for example, provides specialist Toolboxes including capabilities that integrate AI into the complete workflow for developing fully engineered systems [7] and machine learning with supervised and unsupervised learning [8]. The development of AI systems using Python programming is becoming increasingly common [9]. However, whatever system and/or programming language or toolbox is used, the design of a network is fundamental and must be tailored to optimize the application of AI using an ANN for a specific solution. Of specific importance, is both the quality and quantity of the training data. This determines the accuracy of the weights that, along with the design of the network, can be thought of as the 'keys' to the future operation of the network. Thus, there is an obvious application in cryptography where the weights are analogous to the key(s) and the network design, to the encryption algorithm for generating an output cipher that can then be used to encrypt plaintext data.

1.3 Artificial Neural Networks, Data Processing and Deep Learning

As discussed in Section 1.2, it is typical to first of all process the data to generate a feature vector containing metrics that are taken to be a good representation of the essential characteristics of the data, a digital signal, for example. In this way, the number of nodes in the input layer become relatively few compared to the original data, i.e. the length of the digital signal. This is important in regard to utilizing the inevitable limited computational power available to 'drive' an ANN in order to produce an efficient decision-making process (e.g. the computational time required). However, in some cases, it is very difficult to define, in a fully quantitative sense, the elements of the feature vector which are good (unique and unambiguous) characteristics of the data. This problem is often overcome by investigating new properties of the data based on novel analysis methods. For example, texture in an image can be quantified using the principles of fractal geometry and computing metrics such as the Fractal Dimension and multi-fractal parameters. This allows more impressionable features in an image (or the image as a whole), for example, described by the rather elusive term 'texture', to be quantified through Fractal Geometry in Digital Imaging [10] and [11].

The determination of what feature vector should be used, and the elements it contains, determines the size of the feature vector which should ideally be much smaller than the original data from which it has been constructed using a number of different signal or image processing algorithms to output specific metrics. There is a optimality issue that needs to be considered, which is the processing time required to compute the feature vector relative to the time required to train the ANN. However, in recent years, the computational power available through hardware improvements has increased to such a degree, that instead of processing signals to yield a composite feature vector, this process is ignored and the raw data (i.e. of the original digital signal) is used directly to 'feed' the input layer of an ANN. Coupled with many layers consisting of a large number of nodes, this approach significantly increases the complexity of the ANN but reduces that need to decide what signal processing algorithms are required to generate a smaller feature vector *a priori*.

This is the principle associated with a more recent approach to AI known as Deep Learning in which prior signal processing methods are effectively abandoned and high data throughput ANNs designed to process the original data alone. In this context, Figure 2 illustrates the difference between a conventional single hidden layer or 'shallow' ANN and a deep network consisting of three hidden layers.

The principal issue that is driving the deep learning approach is the increase in computing power and efficiency. For example, Google's Inception-V3 is based on using a network consisting of 49 layers with more than 20 million interconnections [13]. But this is of little value unless there is a sufficient amount of training data to compute the millions of weights now required. However, with the increase in computational power coupled with the exponential growth in the Internet of Things (IoT), it is becoming much easier to obtain online access to the training data required. In this sense, deep ANN's are possible because of the growth in the Big Data Society that is now prevalent as well as the computational performance available. The growth in computing power, which is allowing a deep learning paradigm to evolve, is primarily industry driven by companies such as Facebook, Google, and, by national security services where issues such as face recognition and track and trace, for example, are primary concerns. This is possible because super computers from some ten years ago, consisting of thousands of processor units and accommodated in buildings with a large floor space can now be condensed into single Graphical Processing Units (GPUs) required for high resolution gaming, for example.

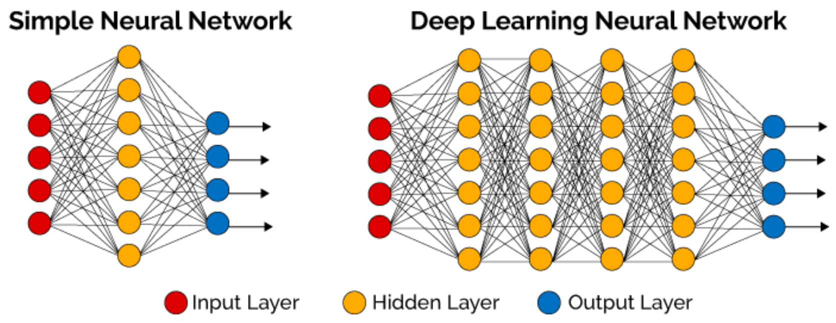


Figure 2. Examples of a single layer 'shallow' neural network (left) that has a 5 -element feature vector with a single layer and a deep neural network consisting of 4 hidden layers, both networks consisting of a 4-node output layer [12].

Irrespective of the computational power available, ANNs require real world data to operate in real world environments. Although this necessity is catered for by the development of access to big data, in some cases, certain data may not be available or is missing from existing data fields. In this case, ANN's can also be of value by training them to create the missing data using other related (real world) data that is available. Thus, not only is AI benefiting from the big data society but can be active in the development, growth and diversity of the big data society (by creating even more data).

Coupled with the training data, the design of an ANN can vary considerably (i.e. the number of layers, the nodes per layer and homogeneity or otherwise of the connections provided etc.) as can the details of the algorithms designed to compute the weights. This includes diffusing the values of the weights over a layer and/or a partition through the layers to a specific depth by applying various spatially invariant filters using the (discrete) convolution operation. These Convolutional Neural Networks (CNNs) were first introduced in the early 1980's. They are a class of deep learning ANNs developed for and applied mainly in the fields of digital image processing and computer vision [14]. In this context, they are replacing many digital image processing algorithms originally developed for pattern recognition using (among other techniques) convolutional filters to segment images into regions of similarity and/or discontinuity (edge detection, for example) [15]. This approach has and continues to be used to develop a set of deterministic, geometric and textural metrics that are traditionally used to design a fuzzy logic decision engine or to train 'shallow learning' ANNs with relatively few inputs and hidden layers.

The application of deep learning solutions to solve complex pattern recognition problems is key to the future development of many technologies and programming environment, functions and systems, all of which are in the process of rapid development. For example, MATLAB have recently developed a new toolbox for solving problems using deep learning [16]. The application of deep learning to cryptography lies beyond the scope of this paper. However, some of the approaches considered in this paper naturally lend themselves to improvements in performance using a deep learning approach. This is especially so in regard to replacing conventional cryptanalysis methods which, in regard to some of the developments presented in this paper, are becoming increasingly redundant [17]. The following section introduces the role ANNs in Cryptography.

1.4 Artificial Neural Networks and Cryptography

One of the principal features of AI is its ability to recognize patterns within complex data. Cryptography relies on maximizing the diffusion and confusion associated with the conversion of plaintext into ciphertext. Ideally, the encrypted data – the ciphertext – should be entirely devoid of any type of pattern. It is required that the ciphertext is a complex randomized representation of a plaintext usually associated with the application of a one-way function which has no inverse solution. This provides the potential for the application of AI in regard to the generation of ciphers, their classification (i.e. their cryptographic strength) and the analysis or cryptanalysis of encrypted data.

Since cryptography relies on the conversion of plaintext into a ciphertext based on the generation of a random number field (the cipher), suppose we input a few initial random numbers to seed a network and then train it using a real source of noise, allowing the weights to be adjusted so that the network outputs an reasonably accurate simulation of the target random data field. In this case, providing the design of the network is known to two communicating entities, Alice and Bob, the initial random numbers and the weights represent the key(s) necessary to reproduce the random number field that can be used to encrypt and then decrypt the data (subject to both the initial random numbers and weights being known to Alice and Bob through application of a key exchange protocol).

The network design (including the computed weights) is equivalent to the encryption/decryption algorithm. In this way, a low number of random elements in the input feature vector can generate a random number field that constitutes the cipher. Such an approach is discussed later on in the paper and compared to another solution to the problem of developing an algorithm for generating ciphers. This is based on the application of a machine learning technique called Evolutionary Computing which can be used to provide a nonlinear equation that, upon iteration, can produce a random number field, the initial condition of the iterator being the (conventional) key.

Just as AI can be used to generate a cipher, the ability for AI to recognize complex patterns lends itself to undertaking the task of recognizing patterns in a ciphertext and thereby looking for signatures that determine a proximity or 'point of attack' where there are weaknesses in the randomized character of the encrypted data. Given that a strong encryption engine should not generate such weaknesses, AI provides a complementary approach to testing the strength of an encryption method, including those methods that are based on the use of AI for generating the encrypted data. Thus, AI has roles to play in association with both data encryption and cryptanalysis. In the latter case, this can be extended to problems associated with information hiding when it is required to detect the signature of a plaintext hidden in another plaintext (Steganalysis) and the signature of a ciphertext hidden in a plaintext (Steganocryptanalysis). In the following section, we briefly review the role of ANNs in Cyber Security.

1.5 Artificial Neural Networks and Cyber Security

Cyber security involves a range of techniques designed to protect the creation, processing, storage and transmission of data over an open network, such as the Internet. A specific and well-known example of this is the infection of files by a virus designed to covertly infiltrate a single computer or a network of computers. Computer viruses have become complex and operate in a stealth mode to avoid detection. New viruses are created each and every day. However, most of these supposedly 'new' viruses are not totally new but are often variations (some of them relative minor variations) on the theme of their predecessors. Many of these viruses just change their form and signatures to avoid detection but their operation and the way they infect files and systems is still the same. This provides the potential for training ANNs on a range of known viruses under the assumption that their digital signatures can be recognized by inspecting future data streams and new files, e.g. [18] and [19].

1.6 About this Paper

Having provided an overview of AI and its applications to Cryptography and Cyber Security in this Section, the paper now provides some specific studies based on previous research already conducted by the authors' coupled with some of their current research themes. Studies are presented which include: (i) the application of Evolutionary Computing to generate unique and unclonable ciphers; (ii) a machine learning strategy for detecting the randomness (or otherwise) of binary streams designed to investigate the potential to attack a ciphertext stream. In the former case, an overview of the authors' current research is given in regard to authenticating documents using a Smartphone as discussed in Section 4.

2 Cryptography using Evolutionary Computing

In Patrick Mahon's secret history of 'Hut 8' – the Naval Section at Bletchley Park (Station X) from 1941-1945 - it is stated that [4]: 'The continuity of breaking Enigma ciphers was undoubtedly an essential factor in our success and it does appear to be true to say that, if a key has been broken regularly for a long time in the past, it is likely to continue to be broken in the future, provided that **no major change in the method of encryption** takes place'. This statement is in regard to the famous Enigma encryption machine used by German armed forces from the mid-1930s until 1945, the underlined component of this statement relating to the design of the enigma machine and not its key settings.

The design of the enigma machine is an example of what today we would call an encryption algorithm whose output is ultra-sensitive to the input key. Today, such encryption algorithms often manifest themselves in terms of just a few lines of source code. They are typically and traditionally based on the application of modular arithmetic coupled with certain mathematical 'tricks' associated with the properties of prime numbers, for example. This includes the famous Rivest, Shamir, Adleman (RSA) algorithm which is arguably the foundation stone for the development of public-private key cryptography whose security relies on the difficulty of factoring large semi-prime numbers into two component prime numbers. Such a factorization has been computationally impossible with conventional digital computing (even with super computers) using procedural algorithms such as Shor's algorithm. However, with the development of quantum computers which allow this algorithm to be implemented efficiently, prime number-based encryption is set to become increasingly obsolete.

Irrespective of the encryption algorithm available (in the past, present or future), there is a conventional rule called the Kerckhoff-Shannon principle which states that a crypto-system (including the algorithm with pre- and post-data processing) should be secure even if everything about the system, except the key(s) is public knowledge. This is stated more succinctly by Claude Shannon as 'the enemy knows the system'. Certainly, it is important that an encryption algorithm can be scrutinized by others in order to confirm (publicly or otherwise) its cryptographic strength. But this is correlated to a certain extent with the relative difficulty of producing new algorithms by hand. AI has overcome such difficulties, and, it is now possible, as is discussed in this section, to produce different encryption algorithms as easily (in principle) as it is to produce different keys. In this context, the encryption algorithm becomes the key, but like a conventional key, still needs to be exchanged between sender and receiver. This allows us to break with the conventional Kerckhoff-Shannon principle. The approach is to apply real world data, in this case, real world noise, which is used to either train an ANN to simulate the noise as briefly discussed in Section 1.4, or, to output a nonlinear function that approximates the noise and can then be cast as an iterator where the initial condition is the (conventional) key. In regard to the latter case, we present an approach based on the application of a machine learning method called Evolutionary Computing.

2.1 Background to the Case

Consider an encryption method that is based on the Gilbert Vernam Cipher [21] which is in fact, a perfectly secure method of encrypting data for one-to-one communications provided a secure method of key exchange is available. The Vernam cipher is a substitution cipher based on generating an array of random numbers to form a vector $\boldsymbol{x} = (x_1, x_2, \dots, x_N)$ – the 'cipher'. This vector represents a digital signal that is taken to be a stochastic field - a purely noise driven signal. The problem is how to generate an algorithm that can be executed on a digital computer to output such a vector that is suitable for encryption using the Vernam cipher or otherwise.

Suppose that some plaintext is written in terms of a set of numbers (using the ASCII, for example), thereby constructing a plaintext vector $\boldsymbol{p}$. The plaintext is typically taken to be the text associated with a natural language, the plaintext vector consisting of decimal integer numbers conforming to the ASCII for the natural language, but, in principle, any code can be used. However, in a more general context, the plaintext vector could consist of elements representing any signal or image, for example. In the latter case, the elements would typically be decimal integers in the range 0-255 for an 8-bit grey level image, but in the former case, the elements may be floating point numbers as can the elements of the cipher. Either way, for a substitution cipher, we can generate the ciphertext by simply adding the two vectors together to generate the ciphertext $\boldsymbol{c} = \boldsymbol{x} + \boldsymbol{p}$. In the case when the plaintext is natural language based text data, coded using 7-bit ASCII and composed of N elements (when the cipher is also composed of N integer values), we can construct the ciphertext as given in Equation (1) below (where mod denotes the modulo operation):

$$c_k = (x_k + p_k) \bmod (127), \; k = 0,1,2,\dots,N-1 \tag{1}$$

The plaintext is then recovered from the decryption equation $p_k = (c_k - x_k) \bmod (127)$, which requires $\boldsymbol{x}$ to be known of course. However, another way of implementing the method of encryption is to write the integer plaintext and cipher vectors as binary strings (using ASCII, for example). In binary space, the ciphertext is then given by $\boldsymbol{c} = \boldsymbol{x} \oplus \boldsymbol{p}$ where $\oplus$ denotes the binary exclusive OR (XOR) operator, the decrypt being given by $\boldsymbol{p} = \boldsymbol{x} \oplus \boldsymbol{c}$**.** In this case, the vector notation used to denote the binary space data $\boldsymbol{c}$, $\boldsymbol{x}$ and $\boldsymbol{p}$ denotes binary strings that are of a finite length $L>N$ where typically, $L>>N$ for a standard plaintext massage.

Whatever the method of encryption that is implemented, a principal issue is how to design an algorithm or a class of algorithms that output a cipher with properties that are consistent with strong encryption. These properties include ensuring that $\boldsymbol{x}$ is statistically unbiased so that the histogram of the cipher is uniformly distributed and equally so, has a power spectral density function that is uniform. Most cipher generating algorithms are based on iterations in which the initial value is the key. They produce pseudo random number streams for which certain critical conditions are required to be met. These conditions include: (i) ensuring that the algorithm generates random numbers that are equally and uniformly distributed irrespective of the key that is used; (ii) given that the cycle length of any finite state computations is itself finite, is the characteristic cycle length of the iteration longer than the length of the plaintexts that are to be used, thereby avoiding patterns in the random number stream that are correlated cyclically.

There are numerous cipher-generating algorithms based on the design of Pseudo Random Number Generators (PRNGs) that have been developed. They tend to fall into three classes which generate: (i) decimal integer random number streams; (ii) floating point random number streams; (iii) random binary streams. The traditional focus has been on the computation of integer streams because of the computational efficiency associated with integer arithmetic. This has typically involved the coupling of modular arithmetic with prime numbers which is why prime number-based cryptography has evolved in the way that it has. By way of an example, the Blum Blum Shub (BBS) cipher [22] is a PRNG first proposed in 1986 that is based on the iteration (for $k = 0, 1, 2, \ldots, N-1$)

$$x_{k+1} = x_k^2 \bmod (M), M = pq \tag{2}$$

where *p* and *q* are prime numbers and x_0 is the key (the initial condition or 'seed') which is a co-prime to *M* meaning that *p* and *q* are not factors of x_0 and not 1 or 0. This algorithm operates on, and outputs a string of pseudo random values that are decimal integers.

The design of cipher generating algorithms that output decimal integers evolved in parallel with the limited computing power available primarily, the limited processing time associated with floating point array processing. With the more recent development of fast floating-point processors and co-processors and specialist real-time digital signal processors, floating-point cipher generation has been able to exploit the study of chaos to produce a new class of chaotic iterators that are based on non-linear iterations and require high precision floating-point arithmetic to be performed.

An example of such an iterator is the Vurhulst cipher given by

$$x_{k+1} = 4rx_k(1 - x_k), \ r \in (0,1) \tag{3}$$

which has a certain synergy with the BBS PRNG given that both are quadratic iterators. In this case, the initial condition x_0 is a floating-point number between 0 and 1. However, this iteration only provides full chaos when *r*=1 which is prohibitive given that x_k converges and then bifurcates multiple times as *r* approaches 1. For this reason, a modification to Equation (3) can be introduced by considering the iteration [23]

$$x_{k+1} = 4(1+r)\left(1+\frac{1}{r}\right)^r x_k(1-x_k)^r \quad r \in (0,1)$$

A range of values of r can then be used in addition to an initial value $x_0 \in (0,1)$ which extends the iterator to a two-parameter algorithm.

In general, we can consider a generic iterative cipher to be of the form

$$x_{k+1} = f(x_k),\ x_0 \in (0,1) \tag{4}$$

where f is some nonlinear function which may include a parameter or set of parameters whose numerical values (or range of values) need to be establish *a priori* to provide a chaotic cipher $\boldsymbol{x}$. In principle, one could attempt to generate a data base of many such non-linear functions. These functions might be modifications of known chaotic maps developed to investigate specific physical (feedback) models of chaos or simply invented. There are, however, a number of issues that need to be appreciated in order to implement such an approach. These issues are considered below.

2.1.1 One-way functions

In the application of Equation (4) to cryptography, it is assumed by default that f is a one-way function which means that x_0 (the key) cannot be derived from a knowledge x_k. However, for certain functions and under certain conditions, this is not the case. For example, with $r = 1$, Equation (3) has the analytical solution [24]

$$x_k = \sin^2(2^k \sin^{-1}\sqrt{x_0})$$

Thus, for $k = 1$, it is possible to invert this equation to obtain the key x_0 meaning that the key can be obtained from knowledge of the first iteration (or higher order iterations) which is clearly not acceptable in the design of a cipher generating algorithm. This result has a synergy with Equation (2) in so much as this equation can be written in the form

$$x_k = \left[x_0^{2^k \bmod C(M)}\right] \bmod M$$

where C is the Carmichael function [25]. Thus, in order to use an iteration of the type given by Equation (4), the nonlinear function must be proved to be a one-way function and not to have an equivalent analytical solution that is invertible.

2.1.2 Equal Probability of States

What-ever nonlinear function is applied, modified or invented, the distribution of $\boldsymbol{x}$ is rarely uniform and the vector must be post-processed to generate such a distribution. This can be done by windowing all values of the vector that conform to a uniform distribution. However, this wastes data and processing time required to generate it, given that many computed floating-point values may have to be discarded. This is an important issue because in assuming the existence of one-way functions, there can exist probability distributions, which are not uniform and are not even statistically close to a uniform distribution, but are nevertheless, computationally indistinguishable from real-world chaos [26]. Hence, checking for equal probability of the states is fundamental. One way of enforcing this requirement is to implement a partitioning strategy to output a random binary string as illustrated in Figure 3, which shows the histogram for Equation (3) and is uniformly distributed for $x_k \in [0.3,0.7]$ (approximately). A binary string can therefore be created by outputting a '0' when $x_k \in [0.3,0.5]$ and a '1' when $x_k \in (0.5,0.7]$ or visa-versa. In this way, a so-called maximum entropy bit stream cipher is obtained in which the encryption process is undertaken using a standard XOR operation.

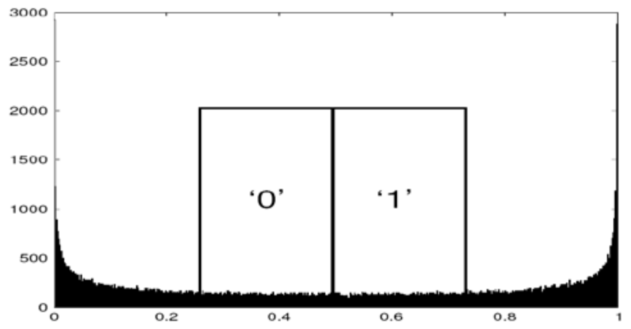


Figure 3. Histogram of the output for a Vurhulst process given by Equation (3) and the partitioning strategy used to generate a maximum entropy binary cipher [24].

Further examples of such ciphers include the functions or maps given in Table 1 which specify the form of the function given in Equation (4). These are examples of stream ciphers obtained by inputting a key $x_0 \in (0,1)$, optimizing their output (through adjusting the value of r) to produce a chaotic number-stream and post processing this output to generate a stream that is uniformly distributed [27].

Table 1. Examples of chaotic maps for the generation of stream ciphers.

Name of Map	**Function**
Generalized Tent Map	$r(1-\|2x-1\|^{\alpha}), \ \alpha > 0$
Quadratic Feedback Map	$rx[1-x(1+x^2)]$
Generalized Sine Map	$\|\sin(\pi r x^{\alpha})\|, \ \alpha > 0$
Tangent Feedback Map	$rx[1-\tan(1/2x)]$
Logarithmic Feedback Map	$rx[1-x\log(1+x)]$

Not all invented or otherwise maps are suitable from a probabilistic point view even though they may exhibit chaos. For example, the map $f(x) = r|1-\tan[\sin(x)]|$, $r = 1.5$ has a highly non-uniform distribution over all probability states and can therefore not be conditioned to produce a maximum entropy cipher using the partitioning strategy illustrated in Figure 3, for example.

2.2 The Lyapunov Exponent

In addition to ensuring equal probability states, it is also necessary to make sure that the nonlinear function is characterized by an iteration in which the Lyapunov exponent is positive, thereby guaranteeing that the iteration is not convergent but chaotic [24]. For a long sequence of numbers $x_1, x_2, \ldots, x_K$, a useful measure of the Lyapunov exponent is given by

$$\lambda(x_0) = \frac{1}{K}\sum_{k=1}^{K}\log|\,f'(x_k)\,| \qquad (5)$$

We require that $\lambda > 1$ for all $x_0 \in (0,1)$. A high, but strictly positive value of the Lyapunov exponent is preferable because the iteration function it is taken to characterize will then generate chaotic trajectories within a few iterations. Thus, ideally what we require is a nonlinear function for which $\lambda \gg 1$ for all

$x_0 \in (0,1)$. In this context, the nonlinear function must also generate an iteration with a long cycle length. In practice, the actual values of the Lyapunov exponent and the cycle length are relative to some known iteration that has been categorized as being cryptographically strong *a priori* with regard of the measures discussed. These issues are difficult to determine analytically and require numerical tests to be undertaken which includes measuring the processing time which cannot be excessive.

2.2.1 Structural Stability

Numerical tests do not guarantee the diffusive properties of a cipher, namely, that the PRNG is structurally stable. Ideally, we require an iteration that has (almost) the same cycle length and Lyapunov exponent for all initial conditions. Most of the known pseudo-chaotic systems do not possess this property and there is no rigorous analytical method, as yet known, for assessing this property. This is an important problem because without solving it, it is not possible to guarantee that a crypto system based on a deterministic chaotic algorithm or set of algorithms will always produce uncorrelated strings for any and all keys, irrespective of their floating-point accuracy.

2.2.2 Computational Unpredictability

Another issue is the determination of unpredictability for chaotic systems, i.e. what properties of a chaotic system grantee its computational unpredictability. This issue is coupled with the problem of algorithmic complexity which cannot be commuted; there is no universal solution for simplifying programs and for proving that the length is minimal. We cannot apply this definition directly to compare the complexity of cryptographic sequences or algorithms. Nevertheless, the theoretical applications are very important. In particular, the Kolmogorov complexity provides a unified approach to the problem of data compressibility [28]. Subject to these important and, as yet, unresolved issues, the applications of 'digital chaos' can yield commercially realizable products but not necessarily products that are scalable in terms of commercial capacity.

Given the issues discussed above in regard to inventing a suitable nonlinear function, it would be beneficial if an approach could be developed where such a function could be evolved from real-world stochastic data, on the understanding that the function would provide only an approximation to the data. To accomplish this, we resort to the use of Evolutionary Computing which is a form of machine learning.

2.3 An Introduction to Evolutionary Computing

In Computer Science, evolutionary computation is known as a family of algorithms for global optimization that are inspired by biological evolution [29]. In practical terms, they are a family of population-based trial and error problem solvers, with a metaheuristic or stochastic optimization character [30], i.e. an initial set of candidate solutions is generated and iteratively updated. Each new generation is produced by stochastically removing less desired solutions, and introducing small random changes. We can say therefore, that evolutionary computational techniques have the capability to produce highly optimized solutions across a wide range of problems. The principles of evolutionary strategies where first introduced in the 1960's [31]. Evolutionary programming was later introduced in the 1990's [32] and while these areas developed separately, by the late 1990's, they were unified as different representatives or 'dialects' of one technology, namely, Evolutionary Computing (EC) [33]. Around the same time, a fourth stream, following the same general concepts had also emerged known as 'genetic programming' [34]. Since then, nature-inspired algorithms have become an increasingly significant part of evolutionary computation.

Many aspects of EC are stochastic, but the starting point of candidate solutions can be either deterministic or stochastic. In EC, genetic algorithms deliver methods to model biological systems that are linked to the theory of dynamical systems, since they are used to predict the future states of the system.

Within the field of EC itself, a software system called 'Eureqa' was one of the first such systems to be developed by the Cornell Creative Machine Laboratory (Cornell University, USA) and then commercialized by *Nutonian Inc*. [35]. The underlying principle is to use genetic programming to generate dynamic equations, each of which provides an increasingly better 'fitness function' to model a given dataset. If this dataset is complex, such as a noise field, the system iteratively generates a sequence of non-linear functions to approximate the input signals [36]. Thus, *Eureqa* is a modeling engine predicated on machine learning, using evolutionary searches to determine an equation that best represents a given data set. *Eureqa* claims to 'automate the heavy lifting inherent in analytics and data science' [37]. The system automatically discovers analytical models requiring almost no human intervention and randomly creates equations via sequences of mathematical building blocks based on a combination of common functions. It is therefore suitable for generating non-linear functions by seeding the system with data from real world noise sources. The functions can then be iterated as required, to produce a key seeded pseudo random number sequence.

Evolutionary Computing is associated with the field of Computational Intelligence, and like AI, involves the process of continuous optimization. AI aims, through iterative processes, to compute a set of optimal weights that determine the flow of information (the amplitude of a signal at a given node) through a network that simulates a simple output subject to a complex input. In this sense, an ANN simulates a high entropy input with the aim of transforming the result into a low entropy output. However, this process can be reversed to generate a high entropy output from a low entropy input. In this sense, ANNs can be used to generate ciphers by simulating natural noise once it has been trained to do so.

2.4 Stream Cipher Generation using Evolutionary Computing

The principal purpose of using EC and ANNs to produce stream ciphers is that the example maps given in Table 1 must derived, modified or just invented subject to an application of the conditions required for them to be compatible with application in cryptography. EC has the potential to do this automatically using real world noise to initiate the evolutionary process. While an ANN approach to generating ciphers is of value in special cases, it does not provide the same flexibility in terms of designing PRNGs using iterated (nonlinear) functions. To do this, an evolutionary algorithms approach is required in which a population-based, stochastic search engine is required that mimics natural selection. Due to their ability to find excellent solutions for difficult and dynamic problems within acceptable computational time, evolutionary algorithms have therefore attracted interest from many areas of science and engineering.

The application of evolutionary computed algorithms to cryptology was first consider in 2013 [38] using *Eureqa* [37]. This system is used to iteratively develop a nonlinear function to describe complex input signals usually associated with experimental data of a chaotic system. If genuine random (delta uncorrelated) noise is input into the system, then from a theoretical point of view, no nonlinear function will be found on an evolutionary basis that models this noise perfectly. Thus, inputting natural noise is a way of 'cheating' the system to 'force' it to provide an approximation to the noise that may be suitable (on an iterative basis) as a PRNG. This suitability is based on a set of tests including an evaluation of the Lyapunov Exponent, checking that the cipher stream is uniformly distributed, as is the Power Spectral

Density Function (PSDF) and so on, as outlined in Diagram 1 (where the ticks indicate that a given test has been passed). By comparison, Diagram 2 shows an equivalent schematic for using an ANN to produce a stream cipher. Comparing the two approaches, it is clear that the weights and the ANN design are equivalent to an evolved function using EC.

In either case, and, in addition to any function being tested against the checks given in Diagrams 1 and 2, the National Institute of Standards (NIST), Computer Security Resource Center, provides a Cryptographic Algorithms Validation Program (CAVP) [39] which includes testing new PRNGs [40]. These NIST tests are an international standard for validating the cryptographic strength of a cipher and are mandatory in the use of a cryptographic algorithm.

Noise Source: Atmospheric Noise, Cosmic Noise, Radioactive Decay, ...

$\downarrow$

Evolutionary Computing System

`http://creativemachines.cornell.edu/eureqa`

$\downarrow$

Iterated Function System $\leftrightarrow$ Approximation to Noise Source

$\downarrow$

Post Processing

$\downarrow$

$$\textbf{Cipher Stream} \longrightarrow \begin{cases} \text{Positive Lyapunov Exponent } \checkmark \\ \text{Uniformly Distributed Statistics } \checkmark \\ \text{Uniformly Distributed PSDF } \checkmark \\ \text{Acceptable Cycle Length } \checkmark \\ \text{Acceptable CPU Time } \checkmark \\ \vdots \end{cases}$$

Diagram 1. Schematic of the processes for evolving a stream cipher using EC.

Noise Source

$\downarrow$

Artificial Neural Network (ANN)

$\downarrow$

Weights $\rightarrow$ ANN $\leftrightarrow$ Approximation to Noise Source

$\downarrow$

Post Processing

$\downarrow$

$$\textbf{Cipher Stream} \longrightarrow \begin{cases} \text{Positive Lyapunov Exponent } \checkmark \\ \text{Uniformly Distributed Statistics } \checkmark \\ \text{Uniformly Distributed PSDF } \checkmark \\ \text{Acceptable Cycle Length } \checkmark \\ \text{Acceptable CPU Time } \checkmark \\ \vdots \end{cases}$$

Diagram 2. Schematic of the processes for generating a stream cipher using an ANN.

2.4.1 Real World Noise Sources

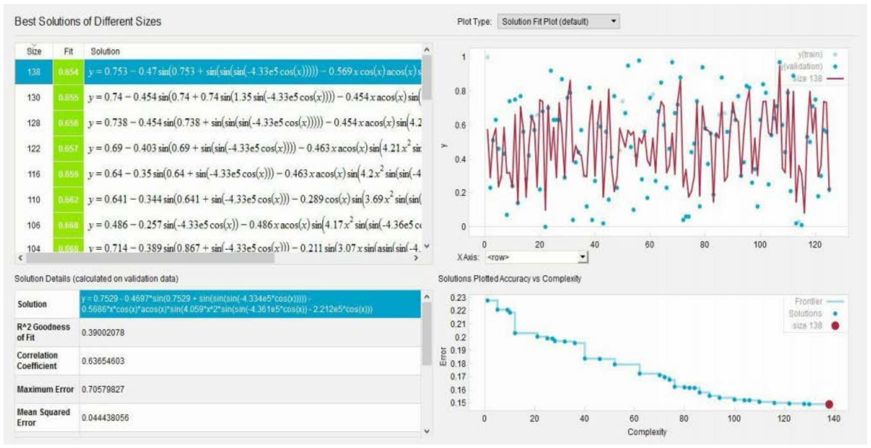
There are a number of real-world noise sources that can be used to input into the systems whose schematics are given in Diagrams 1 and 2. For example, *Random.org* is a free internet resource that provides true random number streams [41]. In this case, the random data is derived from atmospheric noise generated by radio emissions due to lightening; there are approximately 100 lightning strikes to earth per second. Another example is the quantum mechanical noise generated using a reverse biased semiconductor junction. This can be provided in the form of an external USB interface manufactured and supplied by Araneus Information Systems in Finland, for example. Their *Alea* II is a compact true random number generator, also known as a *hardware random number generator*, *non-deterministic random bit generator*, or *entropy source* [42].

2.4.2 Evolved Cipher Generation Examples

Given that this paper is a publication composed for the Transactions of a Society for Science and Education, the authors have chosen to present examples based on previous and current teaching activities associated with the Information and Communications Security course (COS 792) given as part of the Honors program in Computer Science in the Department of Computer Science at the University of Western Cape (UWC), South Africa. One of the course work assignments is as follows: *Read the paper 'Cryptography using Evolutionary Computing' by Blackledge et al. 2013 [38]. Using the information and online references discussed in this paper, develop another example of the cipher given in Section V of this paper using the same approach and online facilities.*

Figure 4 shows a screen shot of *Eureqa* which generated the map given by the following non-linear function:

$$f(x) = 0.7529 - 0.4697\sin(0.7529 + \sin(\sin(\sin(-4.334 \times 10^5 \cos(x)))))$$
$$- 0.569x\cos(x)\,\mathrm{acos}(x)\sin\left(4.059x^2\sin\left(\sin\left(-4.331 \times 10^5\ \cos(x)\right) - 2.212 \times 10^5\cos\left(x\right)\right)\right)$$



Best Solutions of Different Sizes

Size	Fit	Solution
138	0.654	$y = 0.753 - 0.47\sin(0.753 + \sin(\sin(\sin(-4.33e5\cos(x))))) - 0.569x\cos(x)\mathrm{acos}(x)$s
130	0.655	$y = 0.74 - 0.454\sin(0.74 + 0.74\sin(1.35\sin(-4.33e5\cos(x)))) - 0.454x\,\mathrm{acos}(x)\sin($
128	0.656	$y = 0.738 - 0.454\sin(0.738 + \sin(\sin(\sin(-4.33e5\cos(x))))) - 0.454x\,\mathrm{acos}(x)\sin(4.2$
122	0.657	$y = 0.69 - 0.403\sin(0.69 + \sin(\sin(-4.33e5\cos(x)))) - 0.463x\,\mathrm{acos}(x)\sin(4.21x^2\sin$
116	0.659	$y = 0.64 - 0.35\sin(0.64 + \sin(-4.33e5\cos(x))) - 0.463x\,\mathrm{acos}(x)\sin(4.2x^2\sin(\sin(-4$
110	0.662	$y = 0.641 - 0.344\sin(0.641 + \sin(-4.33e5\cos(x))) - 0.289\cos(x)\sin(3.69x^2\sin(\sin($
106	0.668	$y = 0.486 - 0.257\sin(-4.33e5\cos(x)) - 0.486x\,\mathrm{acos}(x)\sin(4.17x^2\sin(\sin(-4.36e5\,$c
104	0.668	$y = 0.714 - 0.389\sin(0.867 + \sin(-4.33e5\cos(x))) - 0.211\sin(3.07x\sin(\mathrm{asin}(\sin(-4.$

Solution Details (calculated on validation data)

Solution	y = 0.7529 - 0.4697*sin(0.7529 + sin(sin(sin(-4.334e5*cos(x))))) - 0.5686*x*cos(x)*acos(x)*sin(4.059*x^2*sin(sin(-4.361e5*cos(x)) - 2.212e5*cos(x)))
R^2 Goodness of Fit	0.39002078
Correlation Coefficient	0.63654603
Maximum Error	0.70579827
Mean Squared Error	0.044438056

Figure 4. Screen shot of *Eureqa* evolving a nonlinear equation to approximate an input vector consisting of random data obtained from *Random.org*.

obtained by one of the students of the 2018 class [43]. The figure shows the outputs of the system which includes the list of equations obtained post 100 iterations, the solution chosen, its goodness of fit, the correlation coefficient, the maximum error and the mean squared error. The plots given in Figure 4 are of the normalized solution fit to an input vector consisting of 138 random data elements (top-right) and the solutions accuracy (Error) plotted against the complexity (lower-right) as the iterations evolve different equations (of greater complexity but with a lower error). The selected equation had a 50% Stability and 76.2% Maturity and used 95% of the training data obtained from *Random.org* [41].

By repeating the process and using *Eureqa* to generate functions that approximate different random vectors (obtain using different noise sources or different noise fields from the same source), it is possible to generate an unlimited number of key dependent stream ciphers by iterating the function obtained according to Equation (4). Table 2 shows some examples of maps that have been randomly selected from the assignment submissions produced by students of the Information and Communications Security course (COS 792) at UWC from 2016-2019. Each map is named after the student that evolved the function using *Eureqa*. These are examples of many hundreds of such functions that have been evolved by students of the course since 2016.

Table 2. Examples of maps evolved using *Eureqa* by a selection of Honors students at UWC.

Name of Student, Date	Function
D J van Roodt, 01-10-2016	$\text{acosh}(x + x^3 \,\text{asinh}(x-1))\,\text{acosh}\left(x - 1 + x^3\left(6 + \text{atan}\left(x\text{asinh}(x^3)\right)\right)\right)x^3$
G Chibba, 28-11-2017	$0.5076 + 0.1094\sin(1.574x) + 0.1568\sin\left(6.107 + x\log\left(0.4619\sin\left(-0.04284x\sin(-0.03866x)\right)\right)\right)$
J Christians, 29-11-2018	$44.42 + 21.91\sin(3.362 + \cos(47.26\cos(41.49x)) - 32.66x^2 + 20.95\cos\left(4.23 + 53.46\cos(\sin(3.362 + \cos(47.26\cos(41.49x)) - 32.66x^2))^2 \sin(3.362 + \cos(47.26\cos(41.49x)) - 32.66x^2)\right) + \cos(41.49x)$
R Lehata, 30-11-2019	$\cos(1.691 + 2.216x) - 0.9704x\cos(1.807 + 2.03x) - 0.2623\cos\left(1.314\,x\sin(x)\ \cos(\cos(1.807 + 2.03x) - 0.9884\,x\cos(1.807 + 2.03x)) - 5.863 \times 10^5 x\right)$

2.5 Discussion

While the output maps evolved by *Eureqa* should be routinely tested against the NIST CAVP (since the maps are essentially an approximation of real-world noise), it can be expected that they will conform to the basic tests for randomness. Thus, the distribution will, by-default, be uniformly distributed and there is no need to apply a partitioning strategy as illustrated in Figure 3, for example, to generate a maximum entropy cipher. In principle, any evolutionary computed cipher can be iterated according to Equation (4), so that it can be used a number of times for a different key $x_0 \in (0,1)$.

Another approach is to use an EC ciphers once and once only, in which case, iteration is not required. This approach over comes the issue of ensuring that the cipher is structurally stable so that the iterator has (almost) the same cycle and Lyapunov exponents for all initial condition (given that most of the known pseudo-chaotic systems do not possess this property). However, this assumes that a large data-base of such ciphers has been created *a priori* and/or can be created in real time as and when required. But since each cipher requires a real-world noise sequence to generate a map, it is arguable that instead of using EC to evolve a map, the original data might be used for encryption instead, which then obviates the need for EC in the first place. On the other hand, since either the map or the data from which it has been evolved must be exchanged between Alice and Bob, the exchange of a map will be preferable given that maps such as those given in Table 2 require less bits to transmit over an unsecured network using a known key exchange algorithm. In either case, a key exchange algorithm is necessary to exchange the map (once) and then the keys used to generate different initial conditions for each plaintext, or the map alone. In the latter case, the map becomes equivalent to the key for a known algorithm.

In addition to providing a cipher, the method can also be used to generate encryption keys. This has been considered in [44] which uses EC to generate keys for encrypting cloud-bound data with Advanced Encryption Standard (AES-128, 192 and 256), the Data Encryption Standard (DES and 3DES) and a novel cryptosystem called 'Cryptor'. One of the problems faced in cryptography is key management. The key management in this case is addressed using an ANN to learn patterns of the encryption key. Once learnt, the key is then discarded to thwart an attack on the key. The key is then reconstructed by the ANN for the purpose of decryption.

2.5.1 Issues on Practical Implementation

An important issue is that EC generated maps can take a significant amount of CPU time to evolve on a standard personal computer, for example, and once generated, may require a large number of high precision floating point operations to compute. In general, the longer EC is left to evolve a function, the more complex the output map becomes. However, this issue is essentially a hardware development concern, and, like the rise of deep learning using complex and computationally intensive ANNs (as discussed in Section 1.3), provides a way forward to generating an unlimited number of unique and unclonable ciphers without the maps having to be 'designed by hand'. This makes available a solution to encrypting data using one-time pads where an evolutionary computed cipher (and the key used to set the initial condition as required) is used once and once only, after which it is discarded. In this context, a known algorithm attack becomes a legacy of the past and conventional PRNGs such as those listed in [45] may eventually become of historical interest alone.

The purpose of allowing an EC system such as *Eureqa* to iterate for what may be many hours on a conventional CPU, is to evolve a cipher that provides a best approximation to the input noise. However, for applications in cryptography, this may not be required as the output obtained after just a few iterations may provide a 'solution' of the type specified by Equation (4) that generates suitable chaos, thereby significantly reducing the evolutionary computational time required. Further, such maps can be implemented on a complementary basis in which a data base of maps (constructed *a priori*) is accessed so that different maps can be randomly selected and used to encrypt different blocks of data over a randomized window of plaintext – multi-algorithmic cryptography.

Practical cryptography is based on passing known statistical tests available at [40], for example, which is designed to ensure the pseudo-random property of a generator. Pseudo-random sequences are used instead of truly random sequences in most cryptographic applications. This is because, a symmetric crypto system can then focus on the application of a key to initiate an iterative formula, a formula that is known to both Alice and Bob and preferably in the public domain. We have considered a method of designing algorithms for generating pseudo random (chaotic) sequences using truly random strings to evolve an iterator that is taken to be an approximation to these sequences. This approach pays no attention to the algorithmic complexity of the iterator which is one of the main problems in the application of chaos to cryptography. Neither does it consider the structural stability of the iterator or its algorithmic complexity. However, it does provide a practical solution to the problem of developing a large database of PRNGs in the application of personalizing encryption algorithms for strictly 1-to-1 communications or '1-to-Cloud' (encrypted) data storage (e.g. [46], [47] and [48]). In this context, public knowledge of a specific encryption algorithm becomes null and void.

By using EC systems such as *Eureqa* seeded with noise, it is possible to generate a nonlinear map with automated control parameter settings. However, the one-step unpredictability does not guarantee that the output sequence will be unpredictable when an adversary has access to a sufficiently long sequence. In other words, the vast number of samples can, on a theoretical basis at least, lead to unpredictability. With these provisions, EC has the potential for generating an unlimited number of ciphers which can be personalized for users to secure their 'Data on the Cloud', for example. Evolutionary computed algorithms can be published so that the approach conforms to the Kerckhoff-Shannon Principle as required, but in the certain knowledge, that a new set of PRNGs can be developed by any individual interested in doing so. In this context, the method might be viewed as a technical solution to the 'democratization of the cipher bureau'.

2.5.2 Encryption Methods

Maps of the type given in Table 2 output a floating-point vector x_k to a precision that can be controlled by the user, subject to the word length of the floating-point processor that is available. If the cipher is taken to be a floating-point data field then for a 'floated plaintext' p_k, the ciphertext will also be a floating-point field, i.e. $c_k = (x_k + p_k)$. To decrypt the ciphertext, the recipient of the data must have access to a processor that can provide the same floating-point accuracy. If this is not possible, the floating-point field can be rounded to output a stream of integers so that for the 7-bit ASCII, the ciphertext is given by Equation (1) where each component of this equation is a decimal integer, re-scaled as required so that $x_k \in [0,127]$, for example. Alternatively, this rounded cipher stream can be written as a binary stream and the cipher computed using $\boldsymbol{c} = \boldsymbol{x} \oplus \boldsymbol{p}$. These encryption models through which a cipher is used to actually encrypt the data are standard Vernam-type encryption algorithms adapted for different data types. However, other encryption models can be used to encrypt the data as will now be discussed.

Instead of adding the plaintext to the cipher or implementing an XOR operation on the binary string conversions, we can apply the convolution operation to a decimal integer or floating-point array. The ciphertext is given by $\boldsymbol{c} = \boldsymbol{x} \otimes \boldsymbol{p}$ where $\otimes$ denotes the convolution sum and is an example of data diffusion using convolutional encoding. Here, the arrays are taken to be vector arrays but convolutional coding can be applied equally well for arrays or arbitrary dimension, most typically for encrypting images using the two-dimensional convolution sum. Whatever the dimension of the process, decryption requires that the ciphertext is de-convolved to recover the plaintext.

De-convolution is a well-known problem in digital signal and image processing and is often an ill-conditioned problem because the process is unstable and requires to be regularized. However, for cryptography, this problem can be solved by pre-conditioning the cipher with its own power spectrum. In this case, the spectrum of the cipher denoted by X and obtained by taking the Fourier transform of $\boldsymbol{x}$ (which is a complex array with complex conjugate X^*) is replaced with the array $X^*/|X|^2,\ |X|^2 > 0$. Using the convolution theorem, the convolutional coding process $\boldsymbol{c} = \boldsymbol{x} \otimes \boldsymbol{p}$ in data space becomes $C = XP$ in frequency space where C and P are the frequency spectra of $\boldsymbol{c}$ and $\boldsymbol{p}$, repectively. Thus, if we let

$$C = \frac{X^* P}{|X|^2} \tag{6}$$

then the spectrum of the plaintext is given by $P = XC$. In practice, this approach requires application of the Fast Fourier Transform. Also, in practice, the condition that $|X|^2 > 0$ can be satisfied by letting $|X|^2 = 1$ if $|X|^2 = 0$ where the $'0'$ is taken to be a floating point number whose value is below the floating point accuracy available on a given processor. This method has been successfully applied for encrypted information hiding (specifically, image Steganocryptography) – see [49], [50] and [51], for example. One of the more useful properties that convolution-based encryption provides, is that it generates a ciphertext that is data redundant. This facilitates a decrypt subject to a ciphertext with relatively significant data error compared to an XOR process which requires bit error correction algorithms to be applied if the ciphertext incurs errors due to transmission noise, for example.

Another method of encryption is to apply a phase-only process [52], [53]. In this case, the ciphertext is obtain using the equation $c_k = p_k e^{ix_k}$, the decrypt then being given by $p_k = c_k e^{-ix_k}$. This is a process that can be applied in any dimension and in data or frequency space. In data space, however, the ciphertext is complex and therefore increases the data by a factor of 2. On the other hand, the plaintext throughput can be doubled in size by using both real and imaginary components of a complex vector $\boldsymbol{p}$. It is of course possible to intercept the ciphertext, compute $|c_k|$ and then attempt to retrieve the phase from this data. However, the phase retrieval problem is severely ill-posed with no uniformly stable solutions. Moreover, the practicality of implementing phase retrieval algorithms is dependent on the dimension. It is well known that for the two-dimensional case, phase estimation algorithms have been developed to provide approximate solutions. These solutions tend to rely on the data (whose amplitude spectrum is known) being sparse. However, for the one-dimensional case, the phase retrieval problem is ambiguousness, the determination of the phase within the extensive solution set being challenging and only able to be considered under suitable *a priori* assumptions or additional information. This is a consequence of the Fundamental Theorem of Algebra which states that every single-variable polynomial with complex coefficients has at least one complex root. This theorem fails for polynomials of two variables. The inability to factor polynomials of two variables is the reason why the two-dimensional phase retrieval is possible and prevents the one-dimensional phase retrieval problem from being solved. This is because the ability to factor polynomials generates ambiguities where multiple phases correspond to the same data. Thus, any attack associated with attempting to solve the one-dimensional phase retrieval problem will no doubt continue to remain a significant challenge for a cryptanalyst.

2.5.3 Natively Binary Chaos

The methodology discussed in Section 2.3 produces ciphers that are based on floating point iterators. In some cases, the output of such iterators can be converted into binary streams either through necessity (such as with the segmentation scheme illustrated in Figure 3) or through a preference to encrypt in binary space (when developing real-time systems, for example). In the latter case, for example, any algorithm that provides a positive only floating-point cipher $\boldsymbol{x} \in [x_{\min}, x_{\max}]$, say, can be post-processed to generate a cipher $\boldsymbol{x} \in [0,1]$ using the equation

$$\boldsymbol{x} := \frac{\boldsymbol{x} - x_{\min}}{||\boldsymbol{x} - x_{\min}||_{\infty}}$$

The binary string is then obtained by applying a round transformation which rounds each element of $\boldsymbol{x}$ to the nearest integer (in this case, 0 or 1). It is therefore natural to ask whether it is possible to design algorithms that output binary strings directly without the need for conversion from decimal integer form, such algorithms generating natively binary chaotic outputs. However, designing such algorithms is relatively difficult compared to their floating-point counterparts which, as discussed in Section 2.3, are, in the context of using EC, effectively unlimited. On the other hand, obtaining real world binary strings is relatively easy and can be obtained through [41] and [42], for example. This leads to the idea of training an ANN to output random binary streams for a few input (binary) strings (the keys). It also necessitates the importance of obtaining tests that can accurately evaluate the randomness or otherwise of a binary string which is considered in the following section.

A solution to designing an algorithm that can generate natively random binary strings is to apply the new generation of Generative Adversarial Networks (GANs) first considered in 2014 [54]. A GAN is based on using two neural networks which compete with each other in a zero-sum game. For a given training set, the technique learns to generate new data but, crucially, with the same statistics as the training set. This can of course include a binary string in which the distribution of a 0's and 1's is the same.

3 Binary String Analysis using Machine Learning

Just as an ANN can be trained to generate random binary strings, so it can be used to evaluated the randomness (or otherwise) of the string. In principle this could be done using a deep learning strategy as discussed in Section 1.3. However, of all the tests on randomness, the information entropy is a metric that is especially sensitive to the random nature of data in general. It is fundamental to quantifying the meaning of information and hence, the lack of it through, for example, the randomization of a plaintext into a ciphertext with an equal probability of the states. This is why, ciphers that are uniformly distributed are referred to as maximum entropy ciphers. In this section we consider a binary entropy test (a modification thereof) that produces a statistically significant difference between a non-random and a genuinely random binary string and further, can assess the degree to which they are one or the other.

A long standing problem in regard to the analysis of a finite binary string or bit stream (of compact support) is how to tell whether the string is representative of non-random (intelligible) information (involving some form of periodicity, for example) or whether it is the product of an entirely random process or whether it is something in between the two. This problem has applications that include cryptanalysis, quantitative finance, machine learning, artificial intelligence and other forms of signal processing involving the general problem of how to distinguish real noise from information that is embedded in, or is entirely lost, in noise. In cryptanalysis, given that digital communications is based on transmitting bit streams, it is routine to analyze the streams in an attempt to identify a signature which reflects whether the information is encrypted (or otherwise) and if so, the strength of the encrypted signature that is present in the binary data. We now consider the basis for solving this problem using an entropy-based analysis that gives a statistically significant difference between non-random and genuinely random binary strings. The statistical significance is compounded in a marked difference in the distribution of the output signals from which some basic statistical metrics can be computed and used to train an ANN, thereby developing a machine learning strategy to analyze the randomness of a binary stream as it changes in time.

3.1 Information Entropy

In Leo Szilard's 1922 doctoral dissertation and companion landmark paper, he showed how a thought experiment concerned with the second law of thermodynamics called the Maxwell Paradox, could be solved by introducing the concept of information entropy which provides a 'balance' to the paradox associated with a perceived decrease in the Boltzmann entropy [55]. Szilard's original concept on information entropy has become the basis of information theory, showing that there is an increase of $k_B \log_2 2$ units of entropy in any measurement where k_B is the Boltzmann constant. This concept was independently re-invented by Claude Shannon in 1949 [56] (to whom credit is usually but erroneously given) and Andre Kolmogorov and Yakov Sinai, who developed a modified form later on in 1959 [57], [58].

In developing a solution to a paradox in thermodynamics, Szilard introduced an idea that is arguably the single most important icon of the information revolution and the digital age. This is because information entropy provides the key for estimating the (average) minimum number of bits needed to encode a string of symbols, based on the frequency of those symbols.

In a more general context, information is a measure of order, a universal measure applicable to any structure or any system. It quantifies the instructions that are needed to produce a certain organization. We compute the information inherent in any given arrangement from the number of choices we must make to arrive at that particular arrangement among all possible arrangements. Intuitively, the more arrangements that are possible, the more information that is required to achieve a particular arrangement. There are several ways in which one can quantify information but an especially convenient one is in terms of binary choices which is quantified through the binary information entropy.

Consider a digital signal $s_k,\ k = 1, 2 \ldots, K$ composed of K (real) values. Let the discrete probability distribution (constructed from N bins) that a specific value s_k occurs in the signal be $P_n,\ n = 1, 2, \ldots, N$. The information associated with an outcome (i.e. a specific value of s_k with a bin) is then $-\log P_n$ which is a measure of the information required to specify s_k in terms of it being a member of a bin where P_n is the distribution of bins. In this context, the (Shannon) Information Entropy (usually denoted by S), is a measure of the mean (the 'expected value') of the information measure $-\log P_n$ and is given by the dimensionless quantity

$$S = -\sum_{n=1}^{N} P_n \log P_n$$

The higher the entropy of a signal becomes the greater its ambiguity, and, in this context, information entropy is a measure of the unpredictability or randomness of any message contained in the signal. This is typically determined by the noise that distorts the information contained in a signal. In general, the information entropy associated with the transmission of information in a signal tends to increase with time. This is due to the increase in noise with time that distorts any signal as it propagates in time, the sources of this noise being multi-faceted but tending to a Gaussian distribution as a consequence of the Central Limit Theorem.

Instead of considering a digital signal composed of decimal integer or floating-point elements, let us now consider the signal to be a binary string whose elements can take on only two values, namely 0 or 1, which are mutually exclusive. In this case, the signal will have a binary distribution $P_n, n = 1,2$ consisting of just two bins and the Binary Information Entropy (BIE) function, denoted by H, becomes

$$H = -\sum_{n=1}^{2} P_n \log_2 P_n = p\log_2 p - (1-p)\log_2(1-p) \quad \text{Bits} \tag{7}$$

where, if we let p denote the probability of 1 occurring in the binary string, then the probability of obtaining a 0 in the same string is $1-p$. Similarly, if p is taken to denote the probability of 0 occurring in the string, then the probability of obtaining 1 is $1-p$. In either case $0\log_2 0 \equiv 0$.

3.2 Evaluating the Order/Disorder of a Binary String

Given a binary string of a finite length, our problem is to evaluate whether the string is a binary representation of genuine noise or whether it contains intelligible information in terms of it having some degree of determinism. The determinism of such a string could include any natural language that has evolved through use, application and repetition without conscious planning but binary coded in a planned and premeditated way, e.g. the ASCII. The purpose is therefore to establish a method by which a finite binary string of arbitrary length can be compared against another in terms of the relative order and/or disorder of all of its bits. Applying a basic binary entropy test is not sufficient. This is primarily because of its failure to differentiate between binary strings that include periodicity and are therefore not random. For perfectly ordered binary strings whose elements are all equal to 1 or all equal 0 and when p = 0 or p = 1, $H(p) = 0$. On the other hand, when $p = 0.5$, $H(p) = 1$ reflecting maximum irregularity. However, for a string such as 01010101 which has a repeating pattern of 01, $p = 0.5$, $H(p) = 1$ so that in this case, the value of $H(p)$ appears to represent a random binary string when in reality, the sting is periodic and therefore not random at all. Thus, what is required is a binary information entropy-based metric that differentiates more fully than is possible with Equation (7) alone. This issue is further quantified in Table 2, which gives exemplars in regard the order and disorder of some 8- bit binary strings.

Table 3. Some exemplars of order and disorder associated with some short binary strings (from [68]).

Example Binary String	Description	Reason
11111111	Perfectly ordered	All 1's
00000000	Perfectly ordered	All 0's
01010110	Mostly ordered	Mostly 01's
01010101	Regular, not disordered	Repeating 01's
11001100	Regular, not disordered	Repeating 1100's
01011010	Mostly ordered	0101 then 1010
01101011	Somewhat disordered	No apparent pattern
10110101	Somewhat disordered	No apparent pattern

There have been a number of algorithms designed to compute various entropy-based metrics for the determination or measurement on the randomness, regularity, irregularity, order, disorder and entropy for binary and other strings [59], [60]. These include measures that aim to characterize randomness and disorder through the entropy of finite strings such the Approximate Entropy [61], [62], Sample Entropy [63] and Fuzzy Entropy [64]. Such metrics are based on algorithms that can be classified as follows: (i) moving window methods that examine sub-strings of the original; (ii) algorithms which generate a metric based on the entire string length. Applications include basic randomness tests [65], [66] and cryptanalysis [67]. This includes the development of a BiEntropy measure which is based upon a weighted average of the Shannon entropy for all but the last binary derivatives [68].

These metrics are the product of many studies that have been and continue to be undertaken to develop suitable tests and measures based on the computation of a single metric. While desirable computationally, focusing on the use of a single metric for this purpose is restrictive and may be statistically insignificant because of its self-selecting data predication. For this reason, we consider a complementary approach to the problem which is based on the application of the Kullback-Leibler Divergence for a stream of data that yields a statistically significant result as opposed to a single metric. This provides the foundations for the application of a machine learning approach.

3.3 Distribution Analysis using the Kullback-Leibler Divergence

Since randomness is a relative concept, a relative metric is considered which provides a measure of how a binary string compares in some way with a string that is known to be the product of a genuinely random process. Further, this comparison needs to be undertaken on a statistical basis, measuring how one probability distribution associated with the binary string compares to a reference probability distribution in terms of its information content. For the reason, we consider the Kullback-Leibler Divergence or Relative (Binary) Entropy function given by

$$R = -\sum_{n=1}^{2} P_n \log_2\left(\frac{Q_n}{P_n}\right)$$

where P_n is the binary histogram of binary string f_k and Q_n is the binary histogram of some reference binary string g_k, both binary strings being finite and of the same length.

Suppose the string f_k is mostly ordered or regular and not disordered (e.g. a binary string representation of some text from a natural language) and g_k is a random string. We require the metric R to be significantly different in terms of its numerical value to the case when both f_k and g_k are random binary strings. Ideally, what is required is to establish a threshold for the value of R below which f_k can be classified as ordered say, and above which, f_k can be classified as random. This is an example of a binary decision-making process that may not be statistically significant and may not necessarily be able to 'monitor' a transition from f_k being random to non-random. Thus, suppose we consider the relative entropy digital signal given by

$$R_m = -\sum_{n=1}^{2} P_{nm} \log_2\left(\frac{Q_{nm}}{P_{nm}}\right), m = 1, 2, \ldots, M \tag{8}$$

where P_{nm} is the m^{th} binary histogram of the input binary string with $P_{n1} = P_{n2} = \cdots = P_{nM}$ and Q_{nm} is the m^{th} binary histogram of an m^{th} random binary string obtained using a PRNG or is obtained from a real-world random binary string source. To illustrate the characteristics of this relative entropy signal, we consider the following cases (which are referred to as such in regard to presenting the results that follow):

Case (i): P_{nm} is the m^{th} binary histogram of a non-random string;

Case (ii): P_{nm} is the m^{th} binary histogram of a random string.

In the results that follow, the non-random string is obtained by generating the binary representation of the text associated with the abstract of this paper, achieved using an ASCII text to binary converter [69] (with the delimiter set to none). A sequence of random binary arrays are generated using the MATLAB uniform distributed random number generator function *rand* (which returns floating point numbers in the interval [0,1]) and applying a round transformation (to output an array consisting of 0's and 1's), each array having the same length and each array being independent of the another.

Figure 5 shows example signals of the relative entropy given by Equation (8) for Case (i) and Case (ii) above with $M = 8000$ and 100-bin histograms. It is then clear that:

- For Case (i), when f_k, is a non-random string, R_m and has a Gaussian-type distribution.
- For Case (ii), when both strings are random, R_m also has a Gaussian-type distribution.

In both cases, application of the Jargue-Bera test for normality shows that the null hypothesis must be rejected, i.e. the series R_m does not actually conform to a normal distribution even though it appears to have the characteristics of one (as given in Figure 5). Nevertheless, Figure 5 demonstrates the ability for Equation (8) to provide a statistically significant measure in regard to evaluating the randomness or otherwise of a finite binary string. The difference in the mean $\bar{R}$ of the relative entropy signal for the two cases is at least two orders of magnitude and can therefore easily be used to differentiate between a random and non-random binary string. But this is just one of many statistical metrics that may be computed from R_m by computing its *J*-bin histogram $H_j, j = 1, 2, \ldots, J$ and evaluating the r^{th} central moment (the moments about the mean $\bar{R}$) given by

$$\mu_r = \sum_{j=1}^{J} (j - \bar{R})^r H_j$$

providing the variance $(r = 2)$, the skewness $(r = 3)$ and Kurtosis $(r = 4)$, for example, coupled with metrics such as the median and the mode given by $||H_j||_\infty$.

Given the demarcation between a random and non-random binary string using Equation (8), the potential exists to compute further statistical metrics and other parameters based on an analysis of the signatures given in Figure 5. These may include the higher statistical moments and spectral properties of the digital signal R_m, for example, designed to develop a feature vector whose purpose is to provide a multi-class classification using an ANN starting with a simple four component feature vector consisting of the Mean, Standard Deviation, Median and Mode.

Since all data can be expressed as a binary string, irrespective of the code used to do so (i.e. ASCII or otherwise), the approach compounded in utilizing Equation (8) provides a relatively generic method of differentiating between random and non-random binary strings. Unlike other approaches to solving this problem, this approach is based on generating a signal computed using Equation (8) and characterizing the distribution of the signal rather than computing a single metric for a binary string. This allows some common statistical metrics to be used to classify changes to the distribution of the signal R_m and how these changes can form the basis for a machine learning approach using complementary statistical parameters for the signal, coupled with metrics associated with the spectrum of the signal and other transformations. The purpose of this is to provide an analysis of a binary stream that can specify whether it is truly random, intelligible (i.e. the product of some communicable information, for example) or partially random in some way. This provides the ability to continuously monitor binary streams to test for plaintext (intelligible) or encrypted (non-intelligible and random) data.

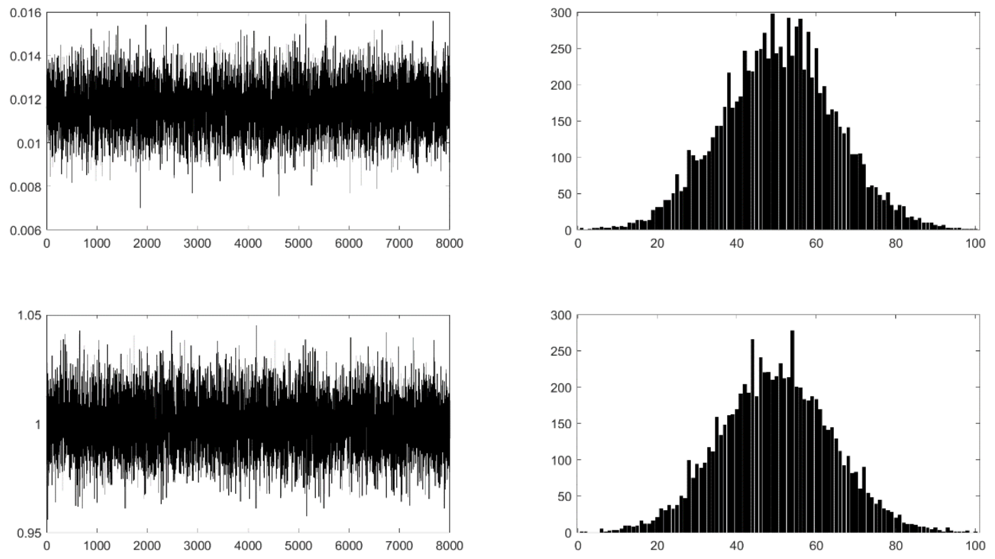


Figure 5. Plots of the R_m (left) given by Equation (8) for $M = 8000$ and the associated 100-bin histograms (right) for Case (i) (above) and Case (ii) (below), respectively.

4 Case Study: Document Authenticity using Unclonable Ciphers

Technologies to make high value documents authentic and impossible or at least difficult to counterfeit have been pioneered for many years. This includes the use of foil holograms, micro-printing, raised print, ultra-violet features, overt and covert watermarks as well as the more recent use of polymers instead of paper for bank notes, for example. With these developments have come a range of specialist readers for checking the authenticity of bank notes at a Point of Sale (PoS) and passports at airports, for example. In this case study, we consider the use of unclonable ciphers based on the application of EC for authenticating documents, specifically with a smartphone. This is because smartphones are now very common with some 3.5B users world-wide (45.12% of the current world population; 2.5B in 2016) and because it is trivial to download a specialist app including those based on the methods discussed here. Consequently, there is no added expense for authentication at a PoS; it is simple to introduce new apps for upgrades and obviates the need to buy specialist readers. There are two potential ways of authenticating a document with a smartphone: (i) using the radio frequency antenna; (ii) using the camera. In this case study we discuss new technologies for both.

4.1 Radio Frequency Identification

Suppose that an Integrated Circuit (IC) coupled to an antenna can be designed that is thin and flexible enough to be embedded into a document and includes enough read only memory to store an unclonable ciphertext that is unique to the IC. In the near field, the antenna of a smartphone can then be used to both activate and read the ciphertext, thereby authenticating the document on-line. The key to this form of RFID (Radio Frequency Identification) [70] is to ensure that the encrypted information maintained on the IC, is unique and unclonable. This requires the generation of numerous chaos-based functions that can be obtained using EC.

FLEX NFC^{TM} stamp antennas [71] are a new generation of ultra-thin (up to 10 µm compared to standard IC's of 75µm or 120µm) low cost and flexible integrated circuits made of amorphous silicon deposited on a polyamide substrate. This extends electronics beyond the realm of conventional mono-crystalline silicon-based IC's. Flexible IC's can be applied to various packaging and security documents, for example, and can be embedded in polymer-based bank notes, for example, using a similar approach to embedding foil holograms. They include a near field communication antenna with typical operational frequencies of 13-14 MHz, a self-resonance frequency of 100 MHz, an impedance of 50-80 Ohms and a reading distance of up to 40 mm.

Authenticity is provided by using physically unclonable maps, a method to create ciphertexts that are inextricably linked to the device. The ciphertexts are unique for a given device, are impossible to read, suspect or eavesdrop on, and any interference by the device destroys them. In addition to EC, this solution can be based on chaos theory which enables the extensions of the dispersion of physical parameters of electronic circuits. The solution's innovation lies in the exploitation of time instability which guarantees that the ciphers and keys are random and repeatable only for a specific device [72]. The solution is implemented as a hardware based non-programmable coprocessor which forms part of the flexible IC and is characterised by a maximum entropy encrypted module with low physical complexity and low power consumption coupled to Internet of Things (IoT).

Figure 6 provides a schematic of the approach for the authentication and identification of a bank note. This is based on correlating a unique ciphertext maintained of a flexible IC (embedded in the note) and the serial number of the bank note. The ciphertext is an encryption of the serial number on the bank note, for example, but other data relating to the note can be included as required. The cipher used for encryption can be an EC generated cipher which is unique to the serial number and unclonable. The ciphertext is read by the antenna of a smartphone (in the near field), transmitted by the phone using the IoT to a secure relational data base where the encrypted data is identified. If identified successfully, the unique cipher is recovered and used to decrypt the data recovering the serial number of the banknote. This result is then transmitted back to the phone quantifying the authenticity or otherwise of the bank note while displaying the serial number as required, and, as illustrated in Figure 7.

In the future development and deployment of this technology, the application of EC based nonlinear functions for generating the many hundreds of millions of unique unclonable ciphers required to embed a unique ciphertext is clear. For any evolved function given in Equation (4), the initial condition can be set to the serial number of the bank note, for example. Each function is 'encoded' in the flexible IC during a production run. Decryption and validation (or otherwise) is accomplished automatically by searching for the unique cyphertext held in a relational data base and its correlation with the serial number upon reception from a smartphone over the IoT.

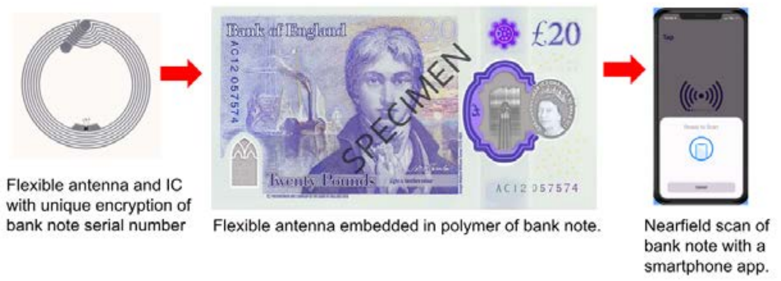
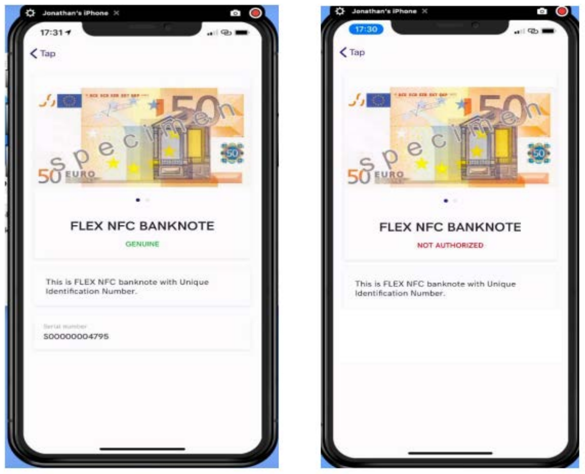

Flexible antenna and IC with unique encryption of bank note serial number
Bank of England
£20
AC12 057574
SPECIMEN
Twenty Pounds
Flexible antenna embedded in polymer of bank note.
Nearfield scan of bank note with a smartphone app.

Figure 6. Schematic illustration of the application to a bank note of a flexible IC with a unique ID number (encryption cipher/key) maintained on a non-programmable coprocessor with a ROM.



This is FLEX NFC banknote with Unique Identification Number.

S00000004795

This is FLEX NFC banknote with Unique Identification Number.

Figure 7. Example of the IoT return for a banknote with an embedded flexible IC using a smartphone to read and transmit the ciphertext and provide the user with a decision on the authenticity of the banknote, i.e. GENUINE (left) or counterfeit and NOT AUTHORISED (right).

4.2 Optical ID

In addition to using online authentication technologies associated with unique RFID's on a smartphone, it is also possible to consider active 'optical solutions' using the digital camera of a smartphone to obtain digital images in ambient light and/or flash light conditions. The retro-reflector method described here is based on the latter case. Compared to RFID, optical ID provides the opportunity to develop a range of optical ciphers and associated identifiers, but this comes at the cost of having to process and transfer digital images using the IoT which requires higher data throughputs than those associated with a RFID. On the other hand, relatively simple print-only and/or additive optical elements and complexes can be included that are simple and cost effective as discussed in this section.

4.2.1 Printed Binary Texture Coding

We consider an approach which is based on the introduction of binary texture codes to a document. Texture codes are based on applying convolutional coding to an image plaintext based on Equation (6) to produce a ciphertext image [51]. The method is relatively robust to distortions of the ciphertext accept in regard to three issues, size, rotation and cropping. In other words, if the ciphertext is printed on a document and then scanned, the scanned image must be cropped correctly, re-oriented (as necessary) and then re-sampled back to the original size of the plaintext image. However, auto-orientation and auto-cropping are routine features of most scanning and remote imaging systems as is the re-sampling of a digital image. Moreover, if the plaintext image is binary, the texture code is data redundant and can therefore can be binarized. Correlation of this binary texture code with the original cipher then recovers the binary plaintext with background noise but with a high signal-to-noise ratio thereby allowing the plaintext to be identified.

Figure 8 shows an example of binary texture coding for a Euro banknote. In this case, the serial number has been used as the binary plaintext, and the binary texture code printed in ultraviolet ink over the entire surface of the banknote at 150 dpi. Under ultraviolet light, an image of the texture code is captured and upon, auto-cropping, auto-correcting (for orientation, as required) and re-sampling the image, the result is decrypted to recover the serial number. In this case, an EC cipher can be used and iterated, according to Equation (4), with double precision and a key equal to the serial number, i.e. $x_0 = 0.00000004795$. The reason for printing the binary texture code using ultraviolet ink is to make the code invisible under visible light. This approach is of course similar to using QR codes, but it provides the basis for incorporating greater information content subject to compatibility with the physical size of the window that is used to print the texture code and the dpi coupled with the resolution of the image capture device.

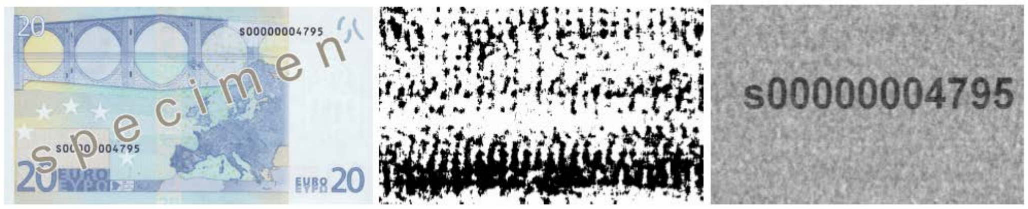


Figure 8. Example of binary texture coding applied to authenticate a bank note. Original note (left), binary texture code (center) and decrypt to recover the serial number (right).

4.2.2 Laser Speckle Analysis

All surfaces have a degree of roughness over different scales and even a highly polished surface has surface roughness at the micro- or nano-scale. The term roughness used here refers to the random undulations on the scale over which the roughness is considered to occur. On the scale of the wavelength of light (400 to 700 nanometres or nm) the roughness of a surface is quite pronounced. This creates internal surface scattering effects. Even with highly polished surfaces such as those required for parabolic mirrors and other optical components, surface irregularities exist on the scale of a wavelength of light. These irregularities are typically measured in terms of the Root Mean Square (RMS) irregularities and surface imperfections (Scratch and Dig) that are typically of the order of tens of nanometres. In the Hubble Space telescope, for example, the primary mirror was polished to give a 0.014-wave RMS wave-front error at 632.8 nm [73]. In addition to maintaining an accurate geometry, the smoothness associated with the surface of optical components on the scale of a light wave is vitally important in order to generate high optical resolutions. The RMS irregularity of a surface determines the characteristics of the Point Spread Function (PSF), in particular the effective bandwidth of the Optical Transfer Function (OTF) - the Fourier transform of the PSF [15]. As the RMS irregularity increases the bandwidth decreases due to surface scattering effects distorting primarily the high frequency band of the OTF.

The scattering effects from a random surface of incoherent light are not phase-sensitive but with coherent light, the phase associated with the scattering processes are observable and measurable and produce a pattern known as a speckle pattern. This is because, with coherent light, the wave-fronts interfere mutually. Speckle patterns are generated by the diffuse reflections of monochromatic light (generated by a laser) which involve multiple scattering events when the scattered optical field can be assumed to be similar to the process of diffusion due to an ensemble of particles undergoing random walks. The diffuse surface reflection of coherent light is generated by any rough surface (i.e. rough on the scale of the wavelength) or media with a complex of scattering objects (with a scale length similar to the wavelength) which have a different permittivity and/or conductivity to the media in which they are suspended. This includes the speckle patterns generated by the surface of paper and other materials used to produce security documents.

The key to using Laser Speckle Analysis for document authentication is to appreciate that the speckle pattern is a signature of the surface topology (on the scale of the wavelength) associated with a chosen surface patch. Moreover, if the physical parameters associated with the incident coherent radiation can be reproduced accurately enough (including wavelength, angle of incidence of the laser beam to the surface, the beam profile and the angle, relative to the surface at which the speckle pattern is recorded) and the surface patch is not deformed over time, then the speckle pattern provides a unique signature of the patch on the document, thereby providing a method of authentication. Further, this method of authentication is passive given that the surface structure of the patch is a natural feature of the material from which the document is composed. All that is required is to decide upon the position and the extent of the surface patch whose speckle pattern signature is desired and then record the pattern produced for a specific optical configuration.

Figure 9 shows a simulated speckle pattern and the characteristic distribution of the pattern plotted on a logarithmic scale. This simulation is obtained using Equation (6) without power spectrum normalisation, i.e. we use $C = X^*P$, where the Plaintext spectrum P is an OTF given by a two-dimensional rectangular function and X is the spectrum of a zero-mean Gaussian distributed random field. The speckle pattern is then given by the absolute square of the inverse Fourier transform of the cipher C which models the scattering cross-section (the intensity of scattered light).

The distribution of the scattering cross-section (the observed laser intensity, $I \geq 0$) associated with a speckle pattern is well known and given by the (negative) exponential distribution

$$P(I) = \lambda \exp(-\lambda I)$$

where λ is the 'rate'. Thus, under a logarithmic transformation,

$$\ln P(I) = \ln\lambda - \lambda I,$$

a relationship that is well demonstrated in the logarithmic scaled histogram given in Figure 9.

By binarizing the speckle pattern, it is possible to generate a code similar to a QR code but one that is produced naturally by the interaction of coherent light with a surface. Such a speckle code can be viewed as a cipher, a cipher that is a unique signature associated with a known surface patch and optical configuration. A speckle pattern can be considered to be another natural noise source in addition to those discussed in Section 2.3.1, for example, and, could be used as an input to the EC systems and approach discussed in Section 2.3. An optical cipher of the type simulated in Figure 9 could be used to encrypt the serial number of a bank note to produce a binary texture code following the approach discussed in Section 4.2.1, for example.

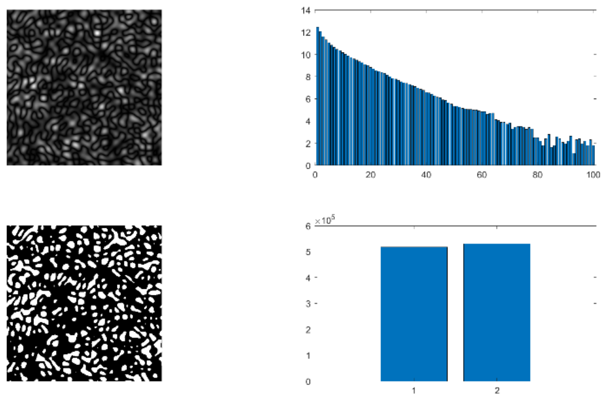


Figure 9. Simulation of a speckle pattern (top-left) and the corresponding grey-level 100-bin histogram (top-right) plotted on a logarithmic scale. A binarization of the speckle pattern designed to produce a matrix of 0's and 1's of near equal population densities is shown in the lower left-hand image. The corresponding 2-bin histogram of this image is given in the lower right-hand side bar graph.

While the methods discussed in this section and that of Section 4.2.1 can be implemented in practice, they both require 'laboratory conditions' to be realised. They require the use of UV or laser light to generate and recover the authentication patterns, patterns that need to be recovered, processed and analysed under controlled conditions with specialist equipment and minimal error in terms of an optical recording configuration. Although the analysis of such patterns for authentication could be considered using Deep Learning, for example, it is clearly not currently possible to implement such an approach in the field using smartphones which are not equipped with UV and laser light sources. Thus, in the following section, another approach is considered that is compatible with the optical facilities provided by a smartphone and can be operated in the field.

4.2.3 Micro-Retro Reflector Texture Coding

Although binary texture codes can be generated without UV printing, a UV print provides a way of hiding the existence of the code under visible ambient lighting conditions. In this way the cipher remains hidden in the optical spectrum. Another way to do this (i.e. introduce a cipher that is invisible under ambient lighting conditions) is to use a random distribution of micro retro-reflectors which is set into the surface of the document providing a random pattern that is unique to that document. Under ambient lighting conditions, the retro-reflector complex will not be observed visually and will not be detected in a captured digital image of a smartphone. However, with a halogen flash that is typically coupled with most modern smartphone cameras, the retro-reflectors will reflect the light and their distribution (or rather the combined effect of that distribution) on the document captured in the digital image that is retained.

Retro-reflectors are devices that reflect light back to its source with minimal scattering [74]. There are many types of such devices including corner reflectors, prism-based reflectors and phase-conjugate mirrors but the most common type makes use of micro retro-reflectors which are each composed of small glass beads or micro-spheres. The beads vary in size from 0.1-1 mm and can be added to paint and lacquers which then cause the surface of the material to which the paint/lacquer has been applied to glow. Highway signs and painted stripes on roadways, for example, have been coated with micro-spherical retro-reflectors for many years. Various reflective sheets are used to coat signs, some of which employ layers of glass beads, while others use sheets embedded with micro-plastic corner prisms or micro-prisms.

Figure 10 shows an example of a bank note which has been coated with *Glowtec* fine grade retro-reflective powder (a pigment for use in spirit and water-based mediums) [75] over a rectangular window within central component of the main graphic. The powder is composed of retro-reflective glass microspheres and has been introduced by adding the powder to a clear nail varnish and then applying the varnish to the surface of the note with the (nail varnish) brush. The size of the microbeads is 30-40 microns each with a refractive index of 1.93 and its application to the surface using a clear lacquer produces a naturally random distribution of microbeads which are unique for any particular note. Under ambient lighting conditions, the existence and distribution of the micro-spheres is not visible and the captured image of the bank-note on a smartphone is 'normal'. However, application of a halogen flash reveals the rectangular region over which the powder doped lacquer has been applied. Thus, by cropping this region of the image, a unique optically generated cipher is obtained which is a combination of the physical micro-bead distribution subject to the resolution at which the image is captured.

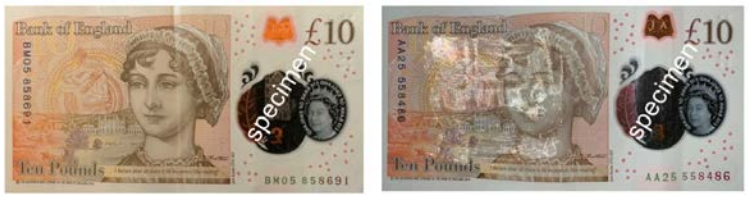
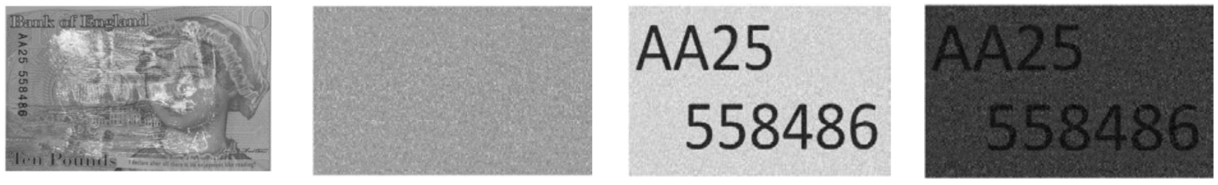


Figure 10. Smartphone images of a banknote with reflective powder applied under ambient lighting conditions (left) and with a LED halogen flash (right). The reflective power, consisting of a randomized complex of micro retro-reflectors, has been applied by doping a clear nail varnish with the powder and 'painting' it onto a rectangular patch that covers the central component of the notes principal graphic as shown on the example on the right-hand-side.

Diffusing a grey level version of this image using Equation (6) with a critical feature of the banknote such as a binary image of the serial number, a ciphertext can be generated and stored in a data base. This ciphertext is then a uniquely encrypted record of the banknotes serial number coupled with the unique optical cipher on the same bank note which remains invisible under ambient lighting conditions. Figure 11 shows example images obtained by applying this process, namely, the cropped optical cipher, the ciphertext for the serial number and a decrypt obtained by re-imaging the cipher. As discussed in Section 4.2.1, the method of encryption by convolutional coding means that the image of the cipher is relatively tolerant to distortion other than orientation, cropping and compatibility with the original digital image size (and the cipher).



(a) (b) (c) (d)

Figure 11. Grey level image of the cropped optical cipher (a), the ciphertext (b) obtained by diffusing the cropped optical cipher with a binary image of the serial number (c) and the decrypt (d). Note that the binary image of the serial number is deliberately corrupted with uniformly distributed noise in order to enhance the uniformity in image diffusivity of the ciphertext shown (a).

The method is also tolerant to changes in the angle at which the image is taken relative to the plane of the banknote. This because the retro-reflectors reflect light back to the source irrespective of the angle of incidence subject to distortions obtained at angles greater than the order of 30 degrees or more.

A typical PoS application of this technique in order to authenticate a bank note is as follows:

- An image of the bank note is taken on a smartphone with the LED flash.
- If the image contains a bright field area (as shown in Figure 10, for example) then the note is taken to include an optical ID; this feature alone being a form of low-level authentication.
- The optical ID is auto-cropped and the cropped image submitted on-line where it is located in a relational database of such images. Note that the crop can include features that lie inside or outside the region in which the optical cipher is placed as long as there is consistency between the crop obtained and that is used to construct the database, the basic principle being to introduce unique randomness into the captured image.
- The cropped image is resampled and decrypted (i.e. a grey level image of the serial number) returned back to the smartphone coupled with image processing as required to enhance the visual quality of the decrypt that is displayed. This is required because the binary plaintext is deliberately corrupted with uniform noise whose purpose of this is to enhance the uniformity of image diffusivity in the ciphertext.
- The decrypt can then be compared with the serial number of the banknote in order to authenticate it.

In this case, the cipher is the image of the retro-reflective powder patch and does not, in principle, require further encrypting. However, in order to strengthen the ciphertext further, it is possible to double encrypt based on extending Equation (6) to the form

$$C = \frac{Y^*}{|Y|^2}\frac{X^*}{|X|^2}P \quad \text{or} \quad C = e^{iY}\frac{X^*}{|X|^2}P$$

where Y is the spectrum of an evolved cipher using EC whose key could also be the serial number of a bank note.

In this application, ANNs will be of value in regard to the automatic recognition of the optical cipher and digital image post-processing that may be required in order to provide a ciphertext that is suitable for decryption. The purpose of this is to allow the recovery of the optical ID to be undertaken in non-ideal conditions associated with the way in which users operate a smartphone subject to critical tolerances. This includes optical clarity, depth and resolution coupled with excess distances from the banknote at which the image is taken, for example, and other issues associated with inevitable mismanagement in the use of an app which is critically dependent on the quality of image capture.

5 Summary, Conclusions and Future Directions

The purpose of this paper has been to provide a broad overview of how AI is starting to be applied in the development of encryption algorithms and cryptanalysis. We have considered the role that Evolutionary Computing can play in providing personal ciphers in the form of unclonable chaotic maps and how, in a complementary way, ANNs can be trained to simulate chaos. In both cases, the training data is based on access to real-world noise sources, which include service providers such as *Random.org*. This approach yields the potential for a 'democratization of the Cipher bureau' where personalized ciphers can be created with relative ease without having to resort to 'hand-crafted' ciphers, many of which are now vulnerable to attack by the implementation of Shor's algorithm [76] on a quantum computer. This is because encryption algorithms that are predicated on the product of large prime numbers can be broken using Shor's algorithm, but until the development of quantum computing, this algorithm was not time computational feasible. In this respect, the approach considered in the paper falls in to the category of Post-quantum Cryptography and the generation of software for prototyping quantum resistant cryptography [77].

5.1 Conclusion

In a broad context, and, in regard to the use of AI for cryptography, there is an interesting analogy that can be made with the work of Station-X at Bletchley Park during the second world war. In this case, the problem was to break the key settings of the Enigma cipher used by the German Armed forces for communications. However, the design of the cipher was known, as was the language being encrypted and many of the expected and repetitive words, phrases and statements (passive Cribs). Moreover, the intercepted Morse coded traffic was known to be encrypted because it was scrambled. Had the traffic been disguised to appear routine through encrypted information hiding (not technically possible at the time), the Morse code have been reconfigured (quite possible at the time), the language been translated to a non-Indo-European language (possible at the time but requiring multi-lingual service providers), repetitive words a phases negated (possible) and the Enigma machine itself changed on a regular basis (not technically possible at the time), then history might have been very different (albeit not necessarily tolerable). It is the last component of this analogy that is the basis for using AI to design ciphers that can be changed regularly or even changed for each communication to give an algorithmic one-time pad. Because this is not yet practicable for all individuals (many of whom may lack the technical IT skills), in the interim, there is ample potential for business opportunities based on establishing on-line service providers to distribute personalized ciphers for 1-to-Cloud applications, 1-to-1 encrypted communication and 1-to-many applications depending on the available distribution infrastructure of an organisation.

5.2 Future Directions

The Binary String Analysis considered in Section 3 provides the basis for the development of a machine learning paradigm based on computing a set of statistical metrics associated with the signal given by Equation (8). This metric set, coupled with other metrics on the spectrum of the signal, for example, is then used to construct a feature vector which forms the input to an ANN. Whether such an approach will be out-ranked by a deep learning approach remains to be identified, but either way, it is important to monitor binary streaming to identify encrypted signatures and structures. In doing so it may be possible to continually monitor internet traffic to estimate the presumed increase in encrypted communications as a result of the approach considered in Section 2 growing in popularity.

The methods introduced in this paper are focused on symmetric encryption in which the key and/or EC cipher are assumed to be exchanged before an encrypted communication is initiated. There are many key exchange methods that can be used for this purpose which exploit asymmetric encryption or a no keys exchange protocol using a three-way pass [53]. Both methods incur weaknesses especially with the evolution of quantum computing that makes legacy systems for key/algorithm exchange based on the RSA algorithm effectively redundant. Asymmetric cryptographic systems are based on trapdoor functions, i.e. functions that have a one-way property unless a secret parameter (trapdoor) is known, and to date, no counterpart of a trapdoor transformation is, as yet, known in chaos theory. The use of EC and other approaches that come under the discipline of AI may be of value in the exploration and possible solutions to this problem.

The applications considered in Section 4 are illustrative of the way in which AI can be used to generate encrypted data for authentication using the IoT coupled with a communications unit which is in widespread use, i.e. the smartphone. The applications presented are typical of the way in which wireless communications, data security and AI are being coupled to secure the transmission of information.

In addition to the optical cipher technique discussed in Section 4.2, it may be of interest to supplement reflective (microspheres) powder with Graphene powder, given that mono-layer Graphene absorbs some 2.3% of incident photons at any wavelength. Thus, by applying Graphene powder with a lacquer to a surface patch of a banknote, for example, it may be possible to use a Graphene based cipher obtained from the absorbance of light rather than, or in addition to, the reflection of the light by retro-reflective powder. Alternatively, it may be possible to dope ink using either retro-reflective powder or Graphene powder subject to the cost effectiveness of such an approach compared to the use of patch ciphers of the type illustrated in Figure 10.

There are of course numerous other applications of machine learning in cryptography than have been considered in this paper [78], [79]. Coupled with complementary developments such as advances in DNA computing [80], deep learning and quantum computing, it can be expected that current encryption algorithms, protocols and standards associated with information and cyber security will change radically in the near future. Moreover, these changes will be under the research, development and control of a much broader global spectrum of society.

Acknowledgements The authors acknowledge the support of the University of KwaZulu-Natal, Technological University Dublin and the Science Foundation Ireland, University of Western Cape, the University of Wales and Warsaw University of Technology. The authors are grateful to Adam Magnuski for his collaboration in regard to the development of the RFID technology discussed in Section 4.1.

REFERENCES

[1] Copeland, B. J., *Artificial intelligence*. Encyclopedia Britannica, 2020. https://www.britannica.com/technology/artificial-intelligence

[2] Turing, A. M., *Computing Machinery and Intelligence*. Mind, 1950, 49: p. 433-460 [Online] Available from: https://www.csee.umbc.edu/courses/471/papers/turing.pdf [Accessed 27 April 2020].

[3] Wikibooks, Artificial Neural Networks/Neural Network Basics, 2020. https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basics

[4] Kostadinov, S., Understanding the Back-Propagation Algorithm, Towards Data Science, 2019. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[5] Capterra, Artificial Intelligence Software, 2020. https://www.capterra.com/artificial-intelligence-software/

[6] AI System Inc., 2020. https://aisystems.ai/

[7] MathWorks, AI with MATLAB, 2020. https://uk.mathworks.com/campaigns/offers/ai-with-matlab.html

[8] MathWorks, Machine Learning with MATLAB, 2020. https://uk.mathworks.com/campaigns/offers/machine-learning-with-matlab.html

[9] Joshi, P., Artificial Intelligence with Python: A Comprehensive Guide to Building Intelligent Apps for Python Beginners and Developers, Packet Publishing Limited, 2017. ISBN: 978-1-78646-439-2. https://www.amazon.com/Artificial-Intelligence-Python-ComprehensiveIntelligent/dp/178646439X

[10] Turner, M. J., Blackledge, J. M. and Andrews, P., Fractal Geometry in Digital Imaging, Academic Press, 1998. ISBN: 0-12-703970-8, 1998.

[11] Fractal Geometry: Mathematical Methods, Algorithms and Applications (Ed. J. M. Blackledge, A. K.

[12] Evans and M. J. Turner), Woodhead Publishing: Series in Mathematics and Applications, 2002. ISBN: 190427500.

[13] Deep Learning in Digital Pathology, Global Engage, 2020. http://www.global-engage.com/life-science/deep-learning-in-digital-pathology/

[14] Google Cloud. AI & Machine Learning Products, Advanced Guide to Inception V3 on Cloud TPU, https://cloud.google.com/tpu/docs/inception-v3-advanced

[15] Zhang, W., Itoh, K., Tanida, J. and Ichioka, Y., Parallel Distributed Processing Model with Local Space-invariant Interconnections and its Optical Architecture, Applied Optics, 1990, 29(32), p. 4790–4797. https://drive.google.com/file/d/0B65v6Wo67Tk5ODRzZmhSR29VeDg/view

[16] Blackledge, J. M., Digital Image Processing, Woodhead Publishing Series in Electronic and Optical Materials, 2005, ISBN-13: 978-1898563495. https://arrow.tudublin.ie/engschelebk/3/

[17] MathWorks, Introducing Deep Learning with MATLAB, 2020. https://uk.mathworks.com/campaigns/offers/deep-learning-with-matlab.html

[18] Maghrebi, H., Portigliatti, T., Prouf, E., Breaking Cryptographic Implementations Using Deep Learning Techniques, Security, Privacy, and Applied Cryptography Engineering (SPACE), 6th International Conference, 2016. [Online] Available from: https://eprint.iacr.org/2016/921.pdf

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

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

[21] Hoare, O., Enigma: *Code Breaking and the Second World War - The True Story through Contemporary Documents*. 2002, Introduced and Selected by Oliver Hoare, UK Public Records.

[22] Vernam, G. S., Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications, Transactions of the American Institute of Electrical Engineers, 1926, 55, p. 109–115.

[23] Blum, L., Blum, M. and Shub, M. A., Simple Unpredictable Pseudo-Random Number Generator, *SIAM* Journal on Computing, 1986, 15 (2), p. 364–383.

[24] Matthews, R., On the Derivation of a 'Chaotic' Encryption Algorithm, Cryptologia, 1984, 8(1), p. 29–41.

[25] Ptitsyn, N. V., Deterministic Chaos in Digital Cryptography, PhD Thesis, De Montfort University, UK and Moscow State Technical University, Russia, 2002.

[26] Erdos, P., Pemereance, C. and Schmutz, E., Carmichael's Lamba Function, Acta Arithmetica, 1991, 58(4), p. 363-385.

[27] Kocare, L., Chaos and Cryptography, 2001, http://rfic.ucsd.edu/chaos/ws2001/kocarev.pdf.

[28] Blackledge, J. M., Ptitsyn, N. V. and Chernenky, V. M., Deterministic Chaos in Digital Cryptography, First IMA Conference on Fractal Geometry: Mathematical Methods, Algorithms and Applications (Eds. J M Blackledge, A K Evans and M Turner), Horwood Publishing Series in Mathematics and Applications, 2002, p. 189-222.

[29] Blackledge, J. M., Cryptography and Steganography: New Algorithms and Applications, Centre for Advanced Studies Text-books, Warsaw University of Technology, ISBN: 978-83-61993-05-6, 2012.

[30] Eiben, A. E. and Schoenauer, M., Evolutionary Computing, Information Processing Letters, 2002, 82(1), p. 1-6.

[31] Ehrenberg, R., Software Scientist: With a Little Data, Eureqa Generates Fundamental Laws of Nature, Science News, 2012, 181(1), p. 20-21.

[32] Rechenberg, I., Evolutionsstrategien, Simulationsmethoden in der Medizin und Biologie. Springer, 1978, 302, p. 83-114.

[33] Fogel, L. J. Owens, A. J. and Walsh, M. J., Artificial Intelligence through Simulated Evolution, Wiley, 1966.

[34] Holland, J. H., Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. University of Michigan Press, 1975.

[35] Eiben, A. E. and Smith, J. E., Introduction to Evolutionary Computing. Springer, 2003, ISBN 978-3-662-05094-1.

[36] Nutonian, Eureqa: The AI-Powered Modeling Engine, 2020. [Online] Available from: www.nutonian.com [Accessed 29 January 2020].

[37] Schmidt, M. and H Lipson, H., Distilling Free-form Natural Laws from Experimental Data, Science, 2009. [Online] Available from: https://science.sciencemag.org/content/324/5923/81

[38] Dubcakova, R., Eureqa: Software Review. Genetic Programming and Evolvable Machines, 2011, 12(2), p. 173-178.

[39] Blackledge, J. M., Bezobrazov, S. Tobin, P. and Zamora, F., Cryptography using Evolutionary Computing, Proc. IET ISSC2013, Letterkenny, Co Donegal, Ireland, June 20-21, 2013 (Awarded Best Paper Prize for ISSC2013). https://arrow.tudublin.ie/aaconmuscon/5/

[40] NIST, Cryptographic Algorithm Validation Program (CAVP), 2020, [Online] Available at https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program

[41] NIST, CAVP Random Number Generators, 2020, [Online] Available at https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program/Random-Number Generators

[42] Random.org, 2020. [Online] Available from https://www.random.org/

[43] Araneus Information Systems Oy, 2020. [Online] Available from https://www.araneus.fi/

[44] Ntolo, S., Student No. 3175732, Course Work Assignment Report, Information and Communications Security (COS 792), Department of Computer Science, University of Western Cape, 2018.

[45] Mosola, N. N., Dlamini, M. T., Blackledge, J. M., Eloff, J. H. P. and Venter, H. S., Chaos-based Encryption Keys and Neural Key-store for Cloud-hosted Data Confidentiality, Southern Africa Telecommunication Networks and Applications Conference (SATNAC), 2017, September 3-10, p. 168-173.

[46] Wikipedia, List of Random Number Generators, 2020. [Online] Available from https://en.wikipedia.org/wiki/List_of_random_number_generators

[47] Blackledge, J. M. and Ptitsyn, N. V., On the Applications of Deterministic Chaos for Encrypting Data on the Cloud, Third International Conference on Evolving Internet, INTERNET 2011, 19-24 June, IARIA, 2011, Luxembourg, p. 78-87. ISBN: 978-1-61208-008-6, 2011.

[48] Tobin, P., Blackledge, J. M., Bezobrasov, S., Personalized Encryptor Generation for the Cloud using Evolutionary Computing, International Workshop on Nonlinear Maps and their Applications (NOMA-2015), Dublin, Ireland, 15-16 June, 2015.

[49] Tobin, P., Blackledge, J. M., McKeever, M., Secrecy and Randomness: Encoding Cloud Data Locally using A One-Time Pad, International Journal on Advances in Security, IARIA, 2017, 10(3), p. 167-181.

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

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

[52] Blackledge, J. M., Tobin, P. Myeza, J. and Adolfo, C. M., Information Hiding with Data Diffusion Using Convolutional Encoding for Super-Encryption, Mathematica Aeterna, 2017, 7(4), p. 319 – 356. https://arrow.tudublin.ie/engscheleart2/127/

[53] Blackledge, J. M., Govere, W. and Sibanda, D., Phase-Only Digital Encryption, IAENG International Journal of Applied Mathematics, 2019, 49(2), p. 212-228. https://arrow.tudublin.ie/engscheleart2/192/

[54] Blackledge, J. M., Tobin, P., Govere, W. and Sibanda D., Phase-only Digital Encryption using a Three-pass Protocol, ISSC2019, IEEE UK and Ireland Signal Processing Chapter, Maynooth University, 17-18, June 2019.

[55] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., Generative Adversarial Networks, Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2014, p. 2672–2680. https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[56] Szilard, L., On the Decrease of Entropy in a Thermodynamics System by the Intervention of Intelligent Beings, Zeitschrift fur Physik, 1922, 53, p. 840–856. http://fab.cba.mit.edu/classes/862.19/notes/computation/Szilard-1929.pdf

[57] Shannon, C. E., A Mathematical Theory of Communication, The Bell System Technical Journal, 1949, 27, p. 379–423, 1948. http://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

[58] A. N. Kolmogorov, A. N., Entropy per Unit Time as a Metric Invariant of Automorphism, Russian Academy of Sciences, 1959, 124, p. 754-755.

[59] Sinai, Y. G., On the Notion of Entropy of a Dynamical System, Russian Academy of Sciences, 1959, 124, p. 768-771.

[60] Marsaglia, G., Random Numbers Fall Mainly in the Planes, Proc. Natl. Acad. Sci., 1968, 61(1), p. 25–28.

[61] Gao, Y., Kontoyiannis, I., Bienenstock, E., Estimating the Entropy of Binary Time Series: Methodology, Some Theory and a Simulation Study, Entropy, 2008, 10, p. 71-99.

[62] Pincus, S., Approximate Entropy as a Measure of System Complexity, Proc. Natn. Acad. Sci., 1991, 88, p. 2297-2301.

[63] Rukhin, A. L., Approximate Entropy for Testing Randomness, J. Appl. Prob., 2000, 37(1), p. 88-100.

[64] Richman, J. S. and Moorman, J. R., Physiological Time-series Analysis using Approximate Entropy and Sample Entropy, American J. of Physiology-Heart and Circulatory Physiology, 2000, 278(6), p. H2039-H2049.

[65] Chen, W., Zhuang, J., Yu W. and Wang, Z., Measuring Complexity using FuzzyEn, ApEn, and SampEn, Med. Eng. and Phys., 2009, 31(1), p. 61-68.

[66] Carroll, J. M., The Binary Derivative Test: Noise Filter, Crypto Aid and Random Number Seed Selector, Simulation, 1989, 53, p. 129-135.

[67] Carroll, J. M. and Sun, Y., The Binary Derivative Test for the Appearance of Randomness and its use as a Noise Filter, Technical Report No. 221, Department of Computer Science, University of Western Ontario, 1998.

[68] Bruwer, C. S., Correlation Attacks on Stream Ciphers using Convolutional Codes, Masters Thesis, University of Pretoria, 2005.

[69] Croll, G. J., BiEntropy – The Approximate Entropy of a Finite Binary String, 2013. https://pdfs.semanticscholar.org/d8d6/bf07050f9dcd026bfafe62c079b4e309dd7a.pdf

[70] Text to Binary translator, Rapid Tables, 2019. [Online] Available from https://www.rapidtables.com/convert/number/ascii-to-binary.html

[71] Radio Frequency Identification (RFID), Wikipedia, 2020, https://en.wikipedia.org/wiki/Radio-frequency_identification

[72] NFC Flex Stamp Antenna, 2020. file://mac/Home/Downloads/L152.pdf

[73] Gołofit, K. and Wieczorek, P. Z., Chaos-based Physical Unclonable Functions, MDPI Applied Sciences, Special Issue on 'Side Channel Attacks', 2019. https://www.mdpi.com/2076-3417/9/5/991

[74] Allen, L. et al., The Hubble Space Telescope Optical Systems Failure Report, NASA, Technical Report, November, 1990. https://ntrs.nasa.gov/search.jsp?R=19910003124

[75] Wikipedia, Retroreflector, 2020. https://en.wikipedia.org/wiki/Retroreflector

[76] Glowtec, Reflective Powder, 2020. https://glowtec.co.uk/Reflective-powder/

[77] Shor, P.W., Algorithms for Quantum Computation: Discrete Logarithms and Factoring, Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, IEEE Comp. Soc. Press, p. 124–134.

[78] Open Quantum Safe Project, Software for Prototyping Quantum Resistant Cryptography, 2016. https://openquantumsafe.org/

[79] Stamp, M., Introduction to Machine Learning with Applications in Information Security, Chapman & Hall/CRC Machine Learning & Pattern Recognition, 2018. ISBN-13: 978-1138626782.

[80] Alan, M. M., Applications of Machine Learning in Cryptography: A Survey, 2019. https://arxiv.org/pdf/1902.04109.pdf

[81] Advances of DNA Computing in Cryptography (Eds. S. Namasudra and G. Chandra Deka), Chapman and Hall/CRC, 2018.