On the Applications of Deterministic Chaos for Encrypting Data on the Cloud

J. M. Blackledge
Information and Communications Security Research Group
Dublin Institute of Technology
url: http://eleceng.dit.ie/icsrg
Email: http://eleceng.dit.ie/blackledge

N. Ptitsyn
Department of Information Processing
and Management Systems
Moscow State Technical University
Email:nptitsyn@gmail.com

***Abstract*—Cloud computing is expected to grow considerably in the future because it has so many advantages with regard to sale and cost, change management, next generation architectures, choice and agility. However, one of the principal concerns for users of the Cloud is lack of control and above all, data security. This paper considers an approach to encrypting information before it is 'placed' on the Cloud where each user has access to their own encryption algorithm, an algorithm that is based on a set of iterated function systems that outputs a chaotic number stream, designed to produce a cryptographically secure cipher. We study cryptographic systems using finite-state approximations to chaos or 'pseudo-chaos' and develop an approach based on the concept of multi-algorithmic cryptography that exploits the properties of pseudo-chaos. Although such algorithms can be taken to be in the public domain in order to conform with the Kerchhoff-Shannon principal, i.e.** *the enemy knows the system*, **their combination can be used to secure data in a way that is unique to each user. This provides the potential for users of the Cloud to upload and transfer data in the knowledge that they are encrypting their data in a way that is algorithm as well key dependent, thereby defeating a known algorithm attack. This paper reports on one application of this approach called** *Crypstic* **in which the encryption engine is mounted on a USB memory stick and where the key is automatically generated by the characteristics of the plaintext/ciphertext file.**

***Keywords*-Cloud computing and Virtualization, Privacy, Security, Ownership and reliability, Data encryption, Deterministic chaos, Multi-algorithmicity**

I. Introduction

Current debates with regard to Cloud Computing assume that little will change for users that depend upon third party hosting for their servers. Further, there appears to be a view that standard security protocols will provide sufficient security in the future. These assumptions ignore the widely held view that the Cloud is insecure. This perception is being constantly reinforced in the mind of the user by the increasingly slow and complicated anti-malware software required and frequent stories in the media about major security breaches - often by hostile governments.

Most businesses rely on some proprietary know-how, process, design or other commercial secret to preserve their competitive position and to try and delay product cycle decay. Business, especially now, is very conscious of the need to avoid fixed and capital costs to reduce their vulnerability to volatility. Cloud Computing, as a capital and fixed cost free approach, is an obvious solution but perceived lack of security for commercially sensitive data is a major barrier to conversion from in-house information and communications technology.

A. The Role of Encryption

Conventional encryption, as a means of securing data, has several drawbacks for commercial users. These include the following: (i) Decision-makers do not understand exactly what encryption is or how to judge the relative strengths of different systems; (ii) Industry certification standards and legal regulations, which are relied upon by both governments and commercial organisations, seek to stratify encryption strength by key length while the underlying algorithms are judged by their resistance to standard attacks. This general approach is common to many industries and is not specific to encryption; (iii) The way in which certification is applied causes, as an unintended consequence, systemic risks to be inherent in any approved system. (iv) Certification is both expensive and slow creating a high barrier to entry for innovative encryption systems and making commercially available systems lag years behind the technologies available to hackers. (v) State regulation with regard to the sale of encryption technology can make the process of commercialising new concepts capital investment intensive [1]; (vi) It is clear that the certification process is valued by governments as a means of understanding, controlling and limiting the strength of encryption to meet their security needs in terms of surveillance. Unfortunately, this approach is fatally flawed as it wrongly assumes that hostile governments do not have equivalent or better capabilities to breach encryption.

B. Data Encryption on the Cloud

How are users going to use The Cloud? For practical purposes, commercial users need to process and store data and communicate with new data and output from stored data. In most cases what is needed is a combination of a Website and Database with secure communications. There is also a need to protect against Malware. This is where encryption faces difficulties as it is impossible to identify Malware if it infiltrates a data stream and is encrypted. Also, in spite of some claims to the contrary, a database cannot be encrypted and then used efficiently. For data to be used it has to be readable. The dilemma therefore is, how can data be securely processed within the cloud. With server hosting, the problem is dealt with by encrypting the communication channel, installing anti-malware, providing physical security and segregating a particular user's servers. Thus, the key is to physically and electronically protect the environment where live processing of data takes place and provide data security using encryption for all communication channels and when data needs to be stored.

The greatest danger from using conventional encryption within The Cloud is that the systemic risks inherent in such encryption methods with only key management to separate secret data contaminate The Cloud as a whole. In other words, a fundamental breach of the encryption engine can bring the whole edifice down. It is this issue that provides the focus for this paper which introduces an approach to encrypting data where all systemic risk can be minimised by replacing the issue of key management with the management of meta-encryption-engines using multiple encryption algorithms based on chaos theory - *multi-algorithmicity*. This is based on a Technology to License called *Crypstic* which is available from Hothouse at Dublin Institute of Technology http://www.dit.ie/hothouse/ and has been developed by the Information and Communications Security Research at the same Institute - http://eleceng.dit.ie/icsrg. The current version is designed specifically for the meta-encryption-engines to be mounted and executed on a USB memory 'key'. However, irrespective of where the engines are mounted, to be credible, their control and processing environment has to be undertaken within a cluster of physically and electronically secure hosting locations.

In the context of using *Crypstic* to secure data on the Cloud, each meta-engine is specific to an individual user and each individual must be properly validated and authorized to have a meta-engine which can be submitted to a user upon request. Each meta-engine device provides a secure entry point to the Cloud so that secure communication to and from the exchange can be achieved without the need for the parties to share their meta-engines. As each meta-engine is seeded for each file or packet differently so that the overall system can act like a 'one-time pad'.

The paper is structured as follows: Section II discusses general issues with regard to the role of encrypting data using chaos before it is uploaded onto the Cloud. Section III discusses the principal issues associated with using chaotic iterators for encrypting which leads to Sections IV and V which focus on the computational issues associated with floating-point approximation and state space partitioning respectively. Section VI is the central kernel of the paper which discusses different chaotic maps including multi-algorithmic maps and introduces some of the principal steps required to design chaotic maps suitable for encrypting data. Section VII presents an implementation of the approach discussed in Section VI and Section VII provides a summary of the work reported in this paper.

II. Cloud Computing and Encryption using Chaos

Cloud computing is set to become a dominating theme in security. The Cloud Security Alliance document *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1* [2] provides an overview of the issues associated with security on the cloud and it is on the basis of this document that we present an approach to encrypting data on the Cloud using chaos.

Cloud computing is inevitable. For example, it avoids the need to acquire infrastructure, it decreases 'time to market' and gives flexibility to update in real time. It is instantly scaleable to meet unexpected increases or decreases in traffic volumes and it saves money by transforming the business model from capital expenditure and depreciation to predictable operating cost. Examples of early adopters to the Cloud include the New York Times who wanted to convert 70 years of articles into PDF format to store it electronically. Using the Cloud it achieved this within 24 hours with no residual unneeded IT infrastructure - a 'one-off' project cost. Start-up companies can use the Cloud to give them full IT capabilities with up-front costs and agility to change requirements and scale up at short notice if successful. The Cloud provides low revenue cost 'Customer Relationship Management' facilities without the need to customize data and process applications. However, there are a number of issues with regard to Cloud Computing which include: trust, loss of privacy, regulatory violation, data replication and erosion of integrity and coherence, application sprawl and dependencies. A general overview of the 'Pros and Cons' associated with Cloud Computing is given in Figure 1.

Of these 'pros and cons', security is a potential major problem for the Cloud. In other words, it is imperative to treat the Cloud as a hostile territory. Consequently user-based security is a likely solution and it is in this context that chaos based cipher generation may provide a solution as discussed in the following section.

A. Chaos Based Cipher Generation

The application of chaos to generating ciphers can create billions of different cryptographically secure encryption engines for users. The commercial solution is to generate a website where users can pay for a unique encryption engine to be produced that, upon a remote payment, can be downloaded and used to encrypt their data before 'storage' on the Cloud. This requires a large database of encryption engines to be created. Once created, a randomly selected sequence of these algorithms can be created on a user-by-user basis. The operational conditions under which this approach can be pursued on a commercial basis depends upon country in

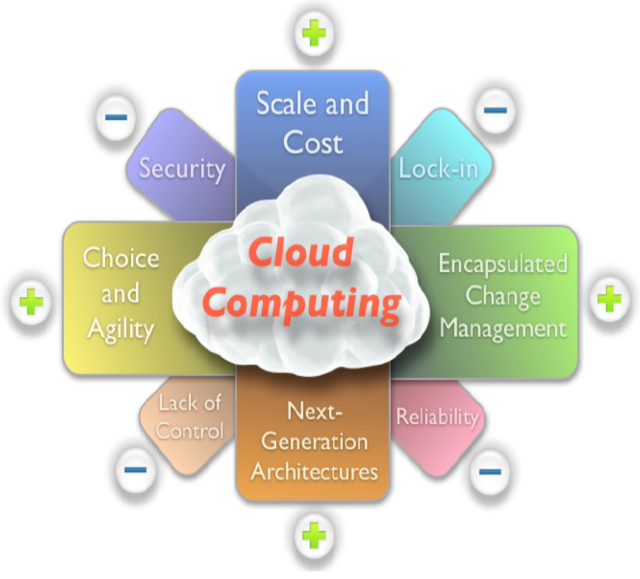

Figure 1. The Pros and Cons associated with Cloud Computing. It should be noted that 'Security' is a 'negative' which relates indirectly to a 'Lack of Control' in terms of possible unauthorised access to data that has been encrypted using standard encryption systems and may therefore be vulnerable to an attack.

which the company is registered. For example, in the UK, commercial operations must conform to the Regulation of Investigatory Powers Act 2000 [1] which inevitably requires an infrastructure to be established involving the employment of staff and is therefore capitalization and overheads intensive.

Chaos can be considered to be a superset of other random number generators used in standard encryption algorithms. There are many disadvantages in using chaos for cryptography but it is nevertheless an interesting application of nonlinear dynamics. The principal value of chaos is the ability to create many different algorithms. This is of course possible with conventional random number generators such as Knuth's M-algorithm [3] but chaos provides greater diversity in terms of the functions available (other than the mod function, for example). However, there are still some major theoretical/computational problems with this approach which include the following:

1) Structurally stable pseudo-chaotic systems: We ideally require a structurally stable cryptosystem, i.e. a system that has (almost) the same cycle length and Lyaponov 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, 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 number streams for any and all keys.

2) Conditions of unpredictability for chaotic systems: What properties of a chaotic system guarantee its computational unpredictability? There is still no theoretically plausible method for evaluating a chaotic system in terms of the necessary/sufficient conditions and properties that will absolutely guarantee the unpredictability of the system to acceptable cryptographic standards. The approach currently being taken is based more on a trial and error approach without the use of an algorithm proving facility. The use of formal methods of software engineering may be of value with regard to this issue.

3) Natively Binary Chaos: While there are, in principle, an unlimited number of chaos based algorithms that can be invented, they currently rely on the use of floating point arithmetic and require high precision FP arithmetic to generate reasonably large cycles (deterministic chaotic algorithm have relatively low cycle lengths which is another disadvantage). These floating point schemes are time consuming given that the number streams they produce are usually converted into bit stream anyway. Designing algorithms that output bit streams directly would therefore be a significant advantage. No theoretical study of this natively binary chaos appears to have been undertaken to date.

4) Asymmetric chaos-based cryptographic: Asymmetric systems are based on trapdoor functions, i.e. functions that have a one-way property unless a secret parameter (trapdoor) is known. One of the best known examples of this is the RSA algorithm that makes use of the properties of prime numbers to design the trapdoor. There is currently no counterpart of a trapdoor transformation, as yet, known in chaos theory.

III. Applications of Chaos for Digital Cryptography

From a theoretical point of view, chaotic systems produce infinite random strings that are asymptotically uncorrelated. However, for applications to digital cryptography, a finite-state systems approach is required which places certain constraints on the design of the algorithm(s). The notion of *pseudo-chaos* introduced in [4], for example, involves a numerical approximation of chaos. The fundamental differences between chaos and pseudo-chaos include the following: (i) the state variable has a finite length (i.e. stores the state with finite precision) and the system has a finite number of states; (ii) the iterated function is evaluated with approximation methods where the result is rounded (or truncated) to a finite precision; (iii) the system may be observed during a finite period of time. The basic problem is that rounding is applied during iteration and the error accumulation causes the original and the approximated processes to diverge. Thus, in general, pseudo-chaos is a poor approximation of chaos because the approximated model does not converge to the original model, and, formally, may exhibit non-chaotic properties including trajectories that eventually become periodic (i.e. contain patterns) and cycles that appear as soon as two states are rounded to the same approximate value. Consequently, the Lyapunov exponent and the Kolmogorov-Sinai information entropy discussed earlier may approach 0. For this reason, it is not possible to directly transform continuous chaotic generators to numerically based generators that require numerical approximations to be made as as summarized in Figure 2. Thus, to use chaos theory for applications in cryptography, a study must be undertaken of pseudo-chaotic systems. This study forms the remit of this paper which is concerned with the question of what are the minimal, typical and maximal periods of the orbits (i.e. string lengths) generated by a pseudo chaotic system?

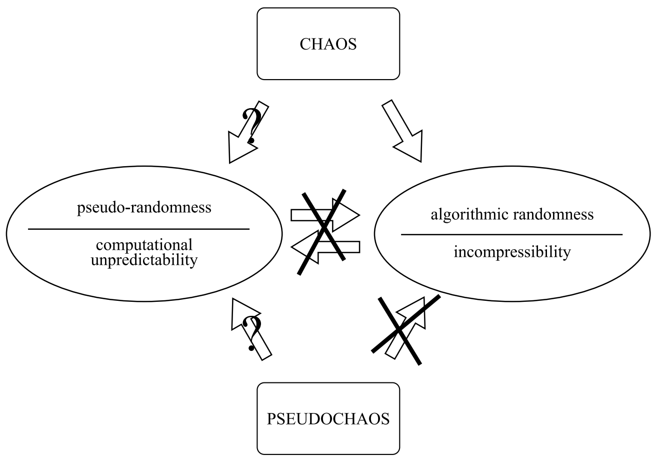

Figure 2. The fundamental properties of chaotic and pseudo-chaotic systems.

Such questions are important in most cryptographic systems. In general, a pseudo-chaotic system produces orbits with different lengths (sometimes called random-length orbits) as illustrated in Figure 3a. Of course, such patterns constitute serious vulnerability as a system may have weak plaintexts and weak keys resulting in recognizable ciphertexts.

If a system has a stable attractor for all initial conditions and parameters, and all orbits have (almost) the same length (Figure 3c), there are more chances to develop a secure encryption scheme. Nevertheless, multiple orbits reduce the search space required for cryptanalysis. An ideal cryptosystem has a single orbit passing through the whole state space (Figure 3b). Another important step in the evaluation of a pseudo-chaotic system is to estimate the Lyapunov exponent of a typical orbit for a time not exceeding its period. However, the analysis of periodic orbits depends critically on the order in which the orbits are considered [5]. Two ordering criteria are considered in the literature, both corresponding to a Lebesgue measure: ordering according to the system size and ordering according to a minimal period or within a period on a lexicographical basis. If the pseudo-chaotic system has a finite precision σ, then the exponential divergence given by

$$e^{n\lambda} = \frac{|f^n(x_0+\varepsilon) - f^n(x_0)|}{\varepsilon}, \qquad n\to\infty, \quad \varepsilon\to 0, \tag{1}$$

will eventually be limited by $\varepsilon = \sigma$. Usually the fraction (1) grows exponentially during the first few iterations and then increases linearly until it finally levels off at a certain finite value.

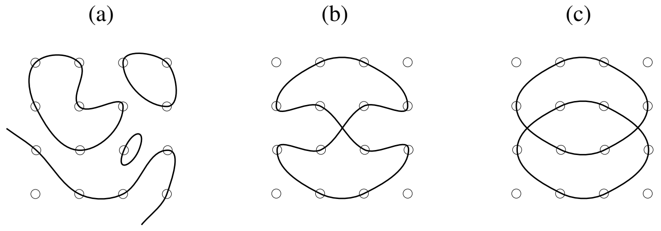

Figure 3. Examples of orbits of a pseudo-chaotic system. (a) Dangerously short orbits (unsuitable for cryptography); (b) A single orbit (the best choice for cryptography); (c) Multiple orbits with the same length (also suitable for encryption).

IV. Floating-point Approximations

Floating-point and fixed point arithmetic are the most straightforward solutions for approximating a continuous system on a finite state machine [6]. Both approaches imply that the state of a continuous system is stored in a program variable with a finite resolution. A state variable x can be written as a binary fraction $b_m b_{m-1}\dots b_1 . a_1 a_2 \dots a_s$, where a_i, b_j are bits, $b_m b_{m-1}\dots b_1$ denotes the integer part and $a_1 a_2 \dots a_s$ is the fractional part of x. Under a finite resolution, instead of $x_{n+1} = f(x)$, we write

$$x_{n+1} = round_k\left(f(x_n)\right),$$

where $k \le s$ and $round_k(x)$ is a rounding function defined as

$$round_k(x) = b_m b_{m-1}\dots b_1 . a_1 a_2 \dots a_{k-1}\left(a_k + a_{k+1}\right).$$

The iterative rounding is accumulative and results in surprisingly different behavior of pseudo-chaos compared with the continuum counterpart. Figure 4 shows how fast the original and approximated trajectories diverge. For cryptographic applications, the rounding off function exposes another danger. Rounding or truncating the state (e.g. to zero values) can lead to the process dropping out of the chaotic attractor and the system state typically remaining at a certain constant value or infinity. Thus, it is necessary to exclude some forbidden initial conditions and parameters which yield short orbits or patterns of behavior after a small number of iterations. Figure 5 is a plot of the average cycle length verses floating-point precision and shows that high precision does not guarantee a sufficiently long trajectory.

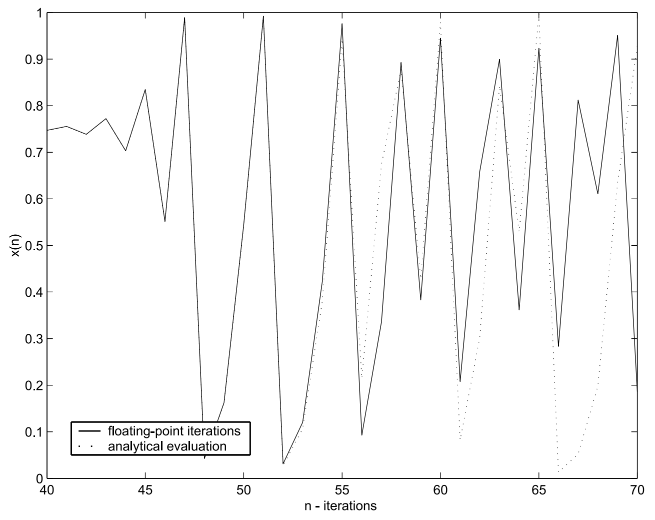

Figure 4. Example trajectories of a continuous-state chaotic system (2) and its 64-bit floating-point approximation. The first curve is obtained by means of the analytical solution (3). The rounding off error is amplified at each iteration and the trajectories diverge exponentially.

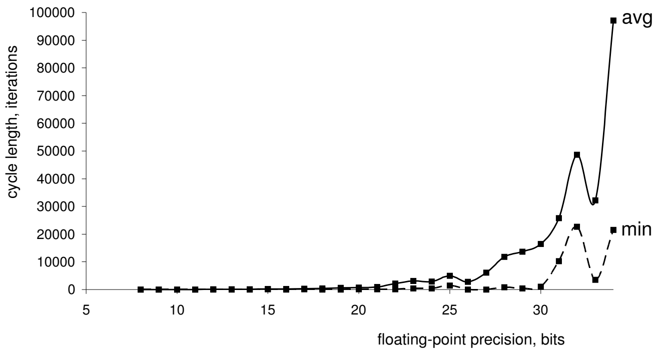

Figure 5. The average (avg) and the minimal (min) cycle length of the logistic system (2) verses floating-point precision obtained from 10 samples of the logistic system.

Another problem associated with the application of pseudo-chaos to encryption is the sensitivity to floating-point processor implementations. Diversified mathematical algorithms or internal precisions in intermediate calculations can lead to a situation where the same encryption application code can generate different cryptographic sequences leading to an incompatibility between software environments. A chaos-based string with two different seeds produces two different sequence with probability 1. This is true for chaotic systems with an infinite state space, where the probability $\Pr\big(f(x_n) = f(x'_n)\big) \to 0$ with $x_n \neq x'_n$ (despite of the fact that f^{-1} is multi-valued). In finite-state approximations, the probability of mapping two points into one is much higher. Furthermore, this can occur at each iteration so that a significant number of trajectories may have identical end routes.

In spite of these shortcomings, a number of investigators have explored the applications of continuous chaos to digital cryptography and in the following sections, an overview of encryption schemes based on a floating-point approximation to chaos is given.

V. Partitioning the State Space

Floating-point cryptographic systems require a mapping from the plaintext alphabet $\{0,1\}^m$ (e.g. 8 bit symbols) to the state space X (e.g. 64 bit floating-point numbers) and, sometimes, from the state space to the ciphertext alphabet. A partition can be defined by a partitioning function $\sigma : X \to \{0,1\}^m$ as with symbolic dynamics. For example, a simple function for two subsets can be designed by taking the last significant bit:

$$\sigma(b_m b_{m-1} \dots b_1 \,.\, a_1 a_2 \dots a_s) = a_s.$$

If a floating-point system is a pseudo-random generator, the function σ must be irreversible as with a hard-core predicate. This can be archived with an equiprobable mapping where partitions are selected in such a way that each symbol occurs with the same probability. However, it is *not* obligatory to cover all the state space or assign symbols to all partitions. On the contrary, we can change the statistical properties of the resulting symbolic trajectory by assigning symbols in a particular way. For example, Figure 6 shows a discrete probability distribution of state points in the attractor of the logistic system. By choosing regions with almost the same probability mass, we obtain better statistics in the output, i.e. avoid any statistical bias associated with a cipher. The number of subsets can be increased, for example, up to 4, 8, 16 etc. In this case the generator will produce more pseudo-random bits per iteration ($m = 2, 3, 4$). However, increasing m reduces the cryptographic strength of the generator since it becomes easier to invert σ.

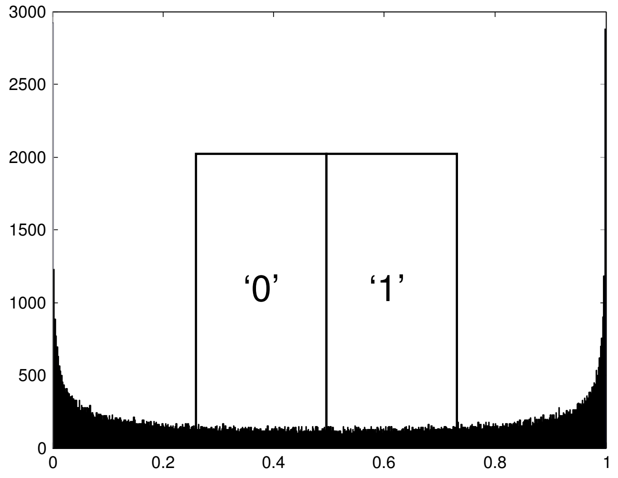

Figure 6. The (discrete) Probability Density Function of a state sequence produced by the logistic system with an incomplete partition.

VI. Example Chaotic Maps

We consider some example chaotic maps which illustrate the principles of using pseudo-chaos for encrypting data.

A. Logistic Map

In 1976, Mitchell Feigenbaum studied the complex behavior of the so-called logistic map given by

$$x_{n+1} = 4rx_n \left(1 - x_n\right), \tag{2}$$

where $x \in (0,1)$ and $r \in (0,1)$. For any long sequence of N numbers generated from the seed x_0 we can calculate the Lyapunov exponent given by

$$\lambda\left(x_0\right) = \frac{1}{N}\sum_{n=1}^{N} \log\left|r\left(1 - 2x_n\right)\right|.$$

For example, the numerical estimation for $r = 0.9$ and $N = 4000$ is $\lambda(0.5) \approx 0.7095$.

With certain values of the parameter r, the generator delivers a sequence, which *appears* pseudo-random. The Freigenbaum diagram (Figure 7) shows the values of x_n on the attractor for each value of the parameter r. As r increases, the number of points in the attractor increases from 1 to 2, 4, 8 and hence to infinity. In this area ($r \rightarrow 1$) it may be considered difficult to estimate the final state of the system (without performing n iterations) given an initial conditions x_0, or vice-versa - to recover x_0 (which can be a key or a plaintext) from x_n. This complexity is regarded as a fundamental advantage in using continuous chaos for cryptography. However, for the boundary value of the control parameter $r = 1$ the analytical solution [7], [8] is:

$$x_n = \sin^2\left(2^n \arcsin\sqrt{x_0}\right). \tag{3}$$

When $n = 1$ we have the initial equation (2). Hence, the state x_n can be computed directly from x_0 without performing n iterations.

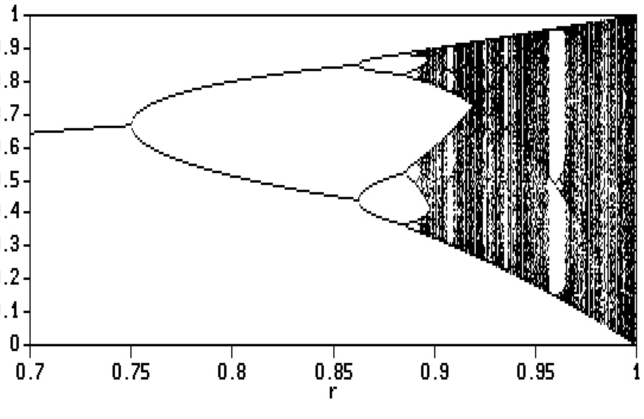

Figure 7. Bifurcation of the logistic map. The most ‘unpredictable’ behavior occurs when $r \rightarrow 1$.

Bianco *et al.* [9] used the logistic map (2) to generate a sequence of floating point numbers which are then converted into a binary sequence. The binary sequence is XOR-ed with the plaintext, as in a one-time pad cipher where the parameter r together with the initial condition x_0 form a secret key. The conversion from floating point numbers to binary values is done by choosing two disjoint interval ranges representing 0 and 1. The ranges are selected in such a way, that the probabilities of occurrence of 0 and 1 are equal (as illustrated in Figure 6). Note, that an equiprobable mapping does not ensure a uniform distribution. Though the numbers of zeros and ones are equal, the order is not necessarily random.

It has been pointed out by Wheeler [10] and Jackson [11] that computer implementations of chaotic systems yield surprisingly different behavior, i.e. it produces very short cycles and trivial patterns (a numeric example in this paper being given in Figure 5).

B. Matthews Map

Matthews [12] generalizes the logistic map with cryptographic constraints and develops a new map to generate a sequence of pseudo-random numbers based on the iteration

$$x_{n+1} = (r+1)\left(\frac{1}{r}+1\right)^r x_n\left(1 - x_n\right)^r, \quad r \in (1,4).$$

The Matthews system exhibits chaotic behavior for parameter values within an extended range (Figure 8) thereby stretching the key space. However, no robust cryptographic system has been created using this map because of the general floating-point issues discussed previously.

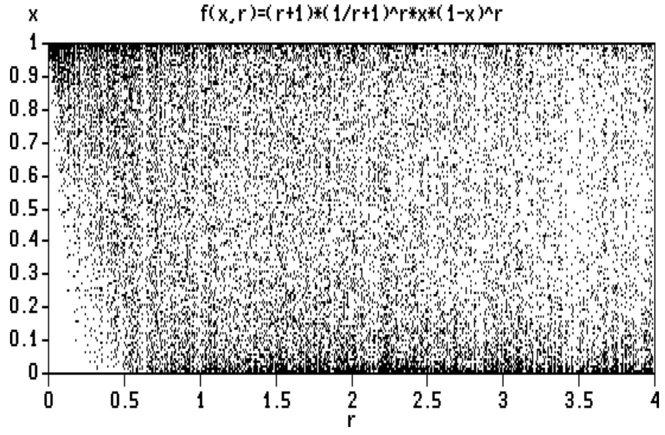

Figure 8. Attractor points corresponding to different values of the parameter r in the Matthews map.

C. Other Examples of Chaotic Maps

Gallagher *et el.* [13] developed a chaotic stream cipher based on the transformation

$$f(x) = \left(a + \frac{1}{x}\right)^{\frac{x}{a}}, \quad x \in (0,10), \quad a \in [0.29, 0.40].$$

Both the initial condition x_0 and the parameter a represent the key. After $n_0 = 200$ iterations, the system encrypts the plaintext byte p_1 into the ciphertext float $c_1 = f^{n_0+n_1}(x_0)$, i.e. the chaotic map is applied $p_1 \in [0, 255]$ times. Subsequent plaintexts are encrypted using the same trajectory. Clearly, the disadvantages of such an encryption scheme are: (i) the data expansion (the floating-point representation of c_i is considerably larger that the source byte p_i; (ii) unstable cycles incident to floating-point chaos generators.

Kotulski [14] proposes a two dimensional map matching the reflection law of a geometric square and defines conditions under which the system is chaotic and mixing. In addition to a range of specific maps suggested by a wealth of authors, there are, in principle, an unlimited number of iteration functions available or that can be invented to generate cryptographic sequences where the nonlinear transformation can be more or less complex, e.g.

$$rx\left[1 - \tan\left(\frac{1}{2}x\right)\right] \quad \text{or} \quad rx\left[1 - \log\left(1 + x\right)\right]$$

Although each system has a particular state distribution in the phase space, qualitatively, its behavior is similar to a basic chaotic system such a logistic map. To increase unpredictability (i.e. the number of states, nonlinearity, complexity) high-order multi-dimensional chaotic system can be used [15]. However, to date, no known systems have been implemented as a working encryption algorithm. This is principally due to the relatively complex numerical integration schemes that are required and the non-uniform distribution of state variables. However, by considering a number of randomly selected pseudo-chaotic algorithms (all of which meet the appropriate design criteria) that operate on randomly selected plaintext blocks, it is possible to produce a multi-algorithmic approach to data encryption which is the principal concept presented in this paper.

D. Pseudo-Chaos and Conventional Cryptosystems

Existing pseudo-random generators can be viewed as pseudo-chaotic systems. For example, consider the Blum-Blum-Shub system [16] given by the iterated function $x_{n+1} = x_n^2 \bmod M$ where $M = pq$, where p, q are two distinct prime numbers each congruent to 3 modulo 4. The output bit b_n is obtained from a predicate $\sigma(x_n)$, which is the last significant bit of x_n. Besides the sensitivity to the initial condition and the topological transitivity, a pseudo-random generator has to be computationally unpredictable. The last property is ensured by a *one-way* iterated function and a hard-core predicate. A one-way transformation is based on a certain mathematical problem, which is considered unsolved. For example, the Blum-Blum-Shub function works under the assumption that integer factorization is intractable. Chaos theory is not focused on the algorithmic complexity of the iterated function, whereas in cryptography the complexity is the key issue, i.e. security.

E. Symmetric Block Ciphers

All classical iterative block ciphers, at least with regard to our notation, are pseudo-chaotic or combinations of several pseudo-chaotic systems. As an example, consider the Rijndael algorithm which form the basis for the Advanced Encryption Standard [17]. The system state x is a two-dimensional array of bits. The plaintext is assigned to the initial conditions x_0 and, after a fixed number of iterations ($n = 10 \ldots 14$), the ciphertext is obtained from the final state x_n. The encryption transformation is a combination of several pseudo-chaotic maps: (i) the substitution phase is a composition of multiplicative inverse and affine transformations; (ii) the mixing phase includes cycle shifts and column multiplication over a finite field; (iii) the round key is obtained from another pseudo-chaotic system.

If we consider the substitution and mixing phases as a single iterated function, the encryption scheme will represent two linked pseudo-chaotic systems (Figure 9).

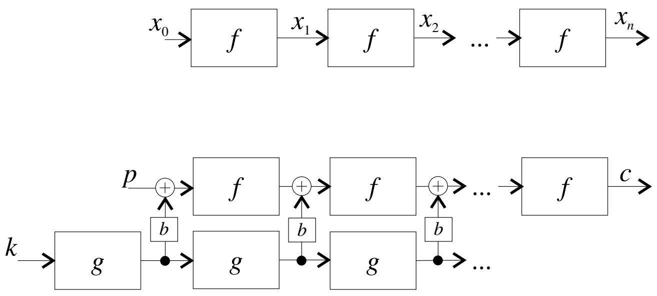

Figure 9. A typical block cipher which is a combination of several pseudo-chaotic systems.

F. Multi-Algorithmic Generators

Protopopescu [18] proposes an encryption scheme based on multiple iterated functions: m different chaotic maps are initialized using a secret key. If the maps depend on parameters, these too are determined by the key. The maps are iterated using floating point arithmetic and m bytes are extracted from their floating point representations, one byte from each map. These m numbers are then combined using an XOR operation. The process is repeated to create a one time pad which is finally XOR-ed with the plaintext. In this paper, we extend the Protopopescu scheme to include a multi-algorithmic approach based on the following properties: (i) Chaotic systems can be connected to each other (i.e. the state of each system influences the states of all other systems) to increase the average orbit length and form a single chaotic system with a large state space and more stable orbits. (ii) The set of chaotic systems (iterated functions) can be different for each encryption session, implemented by supplying an iterated function set with the key. (iii) The output bit can be generated in each q^{th} iteration to increase the independence of bits. (iv) Chaotic systems can be permuted in a complex manner, in particular, the order in which they are utilized or ‘turned on’ by a key.

We can define this extended cryptographic system as

$$\begin{cases} x_{n+1}^1 = f_1(x_n^2, k^1), & b_j^1 = \sigma_1(x_{qj}^1) \\ x_{n+1}^2 = f_2(x_n^2, k^2), & b_j^2 = \sigma_2(x_{qj}^2) \\ \cdots & \cdots \\ x_{n+1}^m = f_m(x_n^m, k^m), & b_j^m = \sigma_m(x_{qj}^m) \end{cases}$$

$$b_j = b_j^1 \oplus b_j^2 \oplus \ldots \oplus b_j^m,$$

where $f_1, f_2, \ldots, f_m$ are iterated functions of the session set, $\langle x_0^1, k^1, x_0^2, k^2, \ldots, x_0^m, k^m \rangle$ are initial conditions, $b_j^1, b_j^2, \ldots b_j^m$ are the internal state bits in the $(n = qj)^{\text{th}}$ moment of time, b_j is the generator output and where the mixing component providing property (i) is given by

$$\begin{cases} x_n^1 = mix_1(x_n^1, x_n^2, \ldots, x_n^m) \\ x_n^2 = mix_2(x_n^1, x_n^2, \ldots, x_n^m) \\ \cdots \\ x_n^m = mix_m(x_n^1, x_n^2, \ldots, x_n^m) \end{cases}$$

A demonstration encryption system - *Crypstic* - based on multiple chaotic systems with extended properties (i)-(iv) is available from [19]. The system solves the problems relating to the floating-point arithmetic to provide $(m-1)$ redundant systems. In practice, an encryption engine can be based on any number of algorithms, each algorithm having been ‘designed’ with respect to the required (maximum entropy) performance conditions through implementation of appropriate conditional parameters T and $\Delta_\pm$ where T is the threshold defining the partition between bits as shown in Figure 6 and $\Delta_\pm$ defines the extent of each partition. The basic steps are as follows:

Step 1: Invent a (non-linear) function f and apply the iteration $x_{n+1} = f(x_n, p_1, p_2, ...)$
Step 2: Normalise the output of the iteration so that $\mathbf{x}_\infty = 1$.
Step 3: Graph the output x_n and adjust parameters $p_1, p_2, ...$ until the output ‘looks’ chaotic.
Step 4: Graph the histogram of the output and observe if there is a significant region of the histogram over which it is ‘flat’.
Step 5: Set the values of the thresholds T and $\Delta_\pm$ based on ‘observations’ made in Step 4.

Analysing of the iteration using a Feigenbaum diagram can also be undertaken but this can be computationally intensive and each function can be categorised in terms of parameters such as the Lyapunov Dimension and information entropy, for example. It should be noted that many such inventions fail to be of practical value because their statistics may not be suitable (e.g. the histogram may not be flat enough or is flat only over a very limited portion of the histogram), chaoticity may not be guaranteed for all values of the seed x_0 between 0 and 1 and the numerical performance of the algorithm may be poor. The aim is to obtain an iteration that is numerically relatively trivial to compute, provides an output that has a broad statistical distribution and is valid for all floating point values of x_0 between 0 and 1.

The functions used for the demo system available at [19] are given in the following table where the values of T, Δ_+ and Δ_- apply to the normalised output stream generated by each function.

Function $f(x)$	r	T	Δ_+	Δ_-
$rx(1-\tan(x/2))$	3.3725	0.5	0.3	0.3
$rx[1-x(1+x^2)]$	3.17	0.5	0.25	0.35
$rx[1-x\log(1+x)]$	2.816	0.6	0.3	0.2
$r(1- \mid 2x-1 \mid^{1.456})$	0.9999	0.5	0.3	0.3
$\mid \sin(\pi r x^{1.09778}) \mid$	0.9990	0.6	0.25	0.25

The functions given in the table above produce outputs that have a relatively broad and smooth histogram which can be made flat by application of the values of T and $\Delta_\pm$ as illustrated in Figure 6 Some functions, however, produce poor characteristic in this respect. For example, the function

$$f(x) = r \mid 1 - \tan(\sin x) \mid, \quad r = 1.5$$

has a highly irregular histogram which is not suitable in terms of applying values of T and $\Delta_\pm$ and, as such, is not an appropriate IFS for this application.

VII. Systems Implementation - *Crypstic*

Crypstic is a generic USB utility for encrypting single data files and can be used as such before a file is uploaded to the Cloud on a file-by-file basis. In conventional encryption systems, it is typical to provide a Graphical User Interface (GUI) with fields for inputting the plaintext and outputting the ciphertext where the name of the output (including file extension) is supplied by the user. *Crypstic* [19] outputs the ciphertext by overwriting the input file. This allows the file name, including the extension, to be used to ‘seed’ the encryption engine and thus requires that the name of the file remains unchanged in order to decrypt. The seed is used to initiate the session key. The file name is converted to an ASCII 7-bit decimal integer stream which is then concatenated and the resulting decimal integer used to seed a hash function whose output is of the form (d, d, f, f, f) where d is a decimal integer and f is a 32-bit precision floating point number between 0 and 1.

The executable file is camouflaged as a *.dll* file which is embedded in a folder containing many such *.dll* files. The reason for this is that the structure of a *.dll* file is close to that of a *.exe* file. Nevertheless, this requires that the source code must be written in such a way that all references to its application are void. This includes all references to the nature of the data processing involved including words such as *Encrypt* and *Decrypt* (strings that are replaced by E and D respectively in a GUI), so that the compiled file, although camouflaged as a *.dll* file, is forensically inert. This must include the development of a run time help facility. Clearly, such criteria are at odds with the 'conventional wisdom' associated with the development of applications but the purpose of this approach is to develop a forensically inert executable file that is obfuscated by the environment in which it is placed. This is based on the forensically inert approach to software engineering.

A. Procedure

The approach to loading the application to encrypt/decrypt a file is based on renaming the *.dll* file to an *.exe* file with a given name as well as the correct extension. Simply renaming a *.dll* file in this way can lead to a possible breach of security by a potential attacker using a key logging system. In order to avoid such an attack, *Crypstic* uses an approach in which the name of the *.dll* file can be renamed to a *.exe* file by using a 'deletion dominant' procedure. For example, suppose the application is called *enigma.exe*, then by generating a *.dll* file called *engine_gmax_index.dll*, renaming can be accomplished by deleting (in the order given) *lld.* followed by *dni_x* followed by *_en* followed by *g* and then inserting a . between *ae* and including *e* after *ex*. A further application is required such that upon closing the application, the *.exe* file is renamed back to its original *.dll* form. This includes ensuring that the time and date stamps associated with the file are not updated.

The procedure described above is an attempt to obfuscate the use of passwords which are increasingly open to attack especially with regard to password protected USB memory sticks. Many manufacturers break all the rules when attempting to implement security. Checking the password and unlocking the stick are two separate processes, both initiated from the PC. Thus, from the point of view of the stick, they are both separate processes, but this is a major flaw. The best USB sticks handle all the encryption to and from the flash memory themselves and do not keep a password at all. The fact that the data cannot be decrypted without it makes it safe. Many USB sticks store a password inside the flash-controller and check it against a password sent by the PC before unlocking the flash-memory. This way, the password cannot be found by reading out the flash-chip manually. Other USB sticks do the same but store the password on flash. Some sticks even store the password on flash and let the PC do the validation.

In addition to the procedures associated with password validation, the concept of password protection is becoming increasingly redundant. For example, Elcomsoft Limited recently filed a US patent for a password cracking technique that relies on the parallel processing capabilities of modern graphics processors. The technique increases the speed of password cracking by a factor of 25 using a GeForce 8800 Ultra graphics card from Nvidia. '*Cracking times can be reduced from days or hours to minutes in some instances and there are plans to introduce the technique into password cracking products*' (http://techreport.com/discussions.x/13460).

B. Protocol

Crypstic is a symmetric encryption system that relies on the user working with a USB memory stick and maintaining a protocol that is consistent with the use of a conventional set of keys, typically located on a key ring. The simplest use of *Crystic* is for a single user to be issued with a *Crypstic* which incorporates an encryption engine that is unique (through the utilisation of a unique set of algorithms). The user can then use the *Crypstic* to encrypt/decrypt files and/or folders (after application of a compression algorithm such as *pkzip*, for example) on a PC before closure of a session. In this way, the user maintains a secure environment using a unique encryption engine with a 'key' that includes a covert access route. If any crypstic, by any party, is lost, then a new pair of sticks are issued with new encryption engines unique to both parties. In addition to a two-party user system, crypstics can be issued to groups of users in a way that provides an appropriate access hierarchy as required.

VIII. Conclusion

There is a fundamental relationship between cryptography and chaos. In both cases, the object of study is a dynamic system that performs an iterative nonlinear transformation of information in an apparently unpredictable but deterministic manner. In terms of sensitivity to initial conditions together with the mixing properties of chaotic systems, with appropriate entropy conscious post-processing, it is possible to ensure cryptographic confusion and diffusion. However, there are a number of conceptual differences between chaos theory and cryptography which include the following: (i) Chaos theory is often concerned with the study of dynamical systems defined on an infinite state space whereas cryptography relies on a finite-state machine and all chaos models implemented on a computer are approximations, i. e. digital computers can only generate pseudo-chaos. (ii) Chaos theory typically studies the asymptotic behaviour of a nonlinear system (i.e. the behaviour of the system as the number of iterations approach infinity when the Lyapunov dimension can be quantified), whereas cryptography focuses on the effect of a small number of iterations that are typically determined by the size of the plaintext. (iii) Chaos theory is not necessarily concerned with the algorithmic complexity but in the interpretation of a physical model from which it has been derived; in cryptography, complexity is the key issue and thus, the concepts of cryptographic security and efficiency have no counterparts in chaos theory. (iv)Classical chaotic systems usually have recognizable attractors whereas in cryptography, we attempt to eliminate any structure by post processing the output to produce a maximum entropy cipher. (v) Unlike chaos in general, cryptographic systems use a combination of independent variables to provide an output that is unpredictable to an observer. (vi) Chaos theory is often associated with the mathematical model used to quantify a physically significant problem, whereas in cryptography, the physical model is of no importance. Point (vi) is of particular importance with regard to the design of chaos based encryption engines. Whereas previous publications in this field (e.g. [12], [9], [20] and [21]) have considered variations on a theme of established chaotic systems, in this paper, we have considered the idea that, in principal, an unlimited number of systems can be 'invented' by a designer in order to provide a limitless range of multi-algorithmic encryption engines.

Cloud computing only represents 4% of current IT spend and is expected to more than double by 2012. Software as a Service (SaaS) by itself is projected to nearly double from \$9B to \$17B (less than 10% of the total market). However, user-security underpins acceptance of cloud architecture. The approach consider in this paper is based on each user having their own encryption engine enabling both protection and control, e.g.

$$\text{PC} + \textit{Crypstic} = \text{Cloud}\ \ \text{Security}$$

The approach to encrypting data discussed in this paper represents a 'paradigm shift' with regard to single algorithm based ciphers that are in the public domain. The importance of this paradigm shift with regard to cryptography in general and, in particular, security on the cloud, may be appreciated in light of the following text taken from Patrick Mahon's secret history of Hut 8 - the naval section at Bletchly Park from 1941-1945 [22]: *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 encypherment takes place*.

Acknowledgment

Jonathan Blackledge is supported by the Science Foundation Ireland Stokes Professorship.

References

[1] Office of Public Sector Information, *Regulation of Investigatory Powers Act 2000*, 2000 CHAPTER 23. http://www.opsi.gov.uk/acts/acts2000/ukpga_20000023_en_1

[2] Cloud Security Alliance http://www.cloudsecurityalliance.org/ Security Guidance for Critical Areas of Focus in Cloud Computing V2.1 http://www.cloudsecurityalliance.org/csaguide.pdf

[3] D. Knuth. The Art of Computer Programming: Volume 2, Seminumerical Algorithms, Second Edition Addison-Wesley, 1981.

[4] B. V. Chirikov and F. Vivaldi. An algorithmic view of pseudorandomness. *Physica D*, (129), 1999.

[5] L. Kocarev. Chaos-based cryptography: a brief overview. *Circuits and systems*, 1(3), 6-21, 2001.

[6] S. Hollasch. Ieee standard 754: floating point numbers, 1998. http://research.microsoft.com/~hollasch/cgindex/coding/ieeefloat.html.

[7] S. Katsura and W. Fukuda. Exactly solvable models showing chaotic bahavior. *Physica*, (130A):597–605, 1985.

[8] J. A. González and R. Pino. Chaotic and stochastic functions. *Physica*, 276A:425–440, 2000.

[9] M. E. Bianco and D. Reed. An encryption system based on chaos theory. US Patent No. 5048086, 1991.

[10] D. D. Wheeler. Problems with chaotic cryptosystems. *Cryptologia*, (12):243–250, 1989.

[11] E. A. Jackson. Perspectives in nonlinear dynamics. Cambridge University Press, 1991.

[12] R. Matthews. On the derivation of a chaotic encryption algorithm. *Cryptologia*, (13):29–42, 1989.

[13] J. B. Gallagher and J. Goldstein. Sensitive dependence cryptography, 1996. http://www.navigo.com/sdc/.

[14] Z. Kotulski and J. Szczepański. Discrete chaotic cryptography. new method for secure communication. In *Proc. NEEDS'97*, 1997. http://www.ippt.gov.pl/~zkotulsk/kreta.pdf.

[15] N. Paar. Robust encryption of data by using nonlinear systems, 1999. http://www.physik.tu-muenchen.de/~npaar/encript.html.

[16] T. Ritter. The efficient generation of cryptographic confusion sequences. *Cryptologia*, (15):81–139, 1991. http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM.

[17] V. Rijmen and J. Daemen. Rijndael algorithm specification, 1999. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.

[18] V. A. Protopopescu, R. T. Santoro, and J. S. Tolliver. Fast and secure encryption-decryption method. US Patent No. 5479513, 1995.

[19] http://eleceng.dit.ie/arg/downloads/crypstic

[20] M. S. Baptista, *Cryptography with Chaos*, Physics Letters A, **240**(1-2), 50-54, 1998.

[21] E. Alvarez, A. Fernandez, P. Garcia, J. Jimenez and A. Marcano, *New Approach to Chaotic Encryption*, Physics Letters A, **263**(4-6), 373-375, 1999.

[22] O. Hoare, Enigma: Code Breaking and the Second World War. The True Story through Contemporary Documents, introduced and selected by Oliver Hoare. UK Public Record Office, Richmond, Surrey, 2002.