2010-01-01

Encryption using Deterministic Chaos

Jonathan Blackledge
Dublin Institute of Technology, jonathan.blackledge59@gmail.com

Nikolai Ptitsyn
Moscow State Technical University, nptitsyn@gmail.com

Recommended Citation

Blackledge, J., Ptitsyn, N.: Encryption using Deterministic Chaos. *ISAST Transactions on Electronics and Signal Processing*, vol. 4, issue 1, pp. 6-17. 2010. doi:10.21427/D7VS65

Encryption using Deterministic Chaos

Jonathan Blackledge
School of Electrical Engineering Systems
Dublin Institute of Technology
Dublin 8, Ireland
url: http://eleceng.dit.ie/blackledge
Email: http://eleceng.dit.ie/blackledge

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

***Abstract*—The concepts of randomness, unpredictability, complexity and entropy form the basis of modern cryptography and a cryptosystem can be interpreted as the design of a key-dependent bijective transformation that is unpredictable to an observer for a given computational resource. For any cryptosystem, including a Pseudo-Random Number Generator (PRNG), encryption algorithm or a key exchange scheme, for example, a cryptanalyst has access to the time series of a dynamic system and knows the PRNG function (the algorithm that is assumed to be based on some iterative process) which is taken to be in the public domain by virtue of the Kerchhoff-Shannon principal, i.e.** *the enemy knows the system*. **However, the time series is not a compact subset of a trajectory (intermediate states are hidden) and the iteration function is taken to include a 'secret parameter' - the 'key'. We can think of the sample as being 'random', 'unpredictable' and 'complex'. What do these properties mean mathematically and how do they relate to chaos? This paper focuses on answers to this question, links these properties to chaotic dynamics and consider the issues associated with designing pseudo-random number generators based on chaotic systems. The theoretical backound associated with using chaos for encryption is introduced with regard to randomness and complexity. A complexity and information theortic approach is considered based on a study of the complexity and entropy measures associated with chaotic systems. A study of pseudo-randomness is then given which provides the foundations for the numerical methods that need to be realed for the practical implementation of data encryption. 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-chaotic algorithms.**

I. Introduction

The concepts of randomness, unpredictability, complexity and entropy form the basis of modern cryptography and a cryptosystem can be interpreted as the design of a key-dependent bijective transformation that is unpredictable to an observer for a given computational resource. In part I of this paper we link these concepts to chaotic dynamics and consider the issues associated with designing pseudo-random number generators based on chaotic systems.

For any cryptosystem, including a Pseudo-Random Number Generator (PRNG), encryption algorithm or a key exchange scheme, for example, a cryptanalyst has access to the time series of a dynamic system and knows the PRNG function (the algorithm assumed to be based on some iterative process) which is taken to be in the public domain by virtue of the Kerchhoff-Shannon principle, i.e. *the enemy knows the system*. However, the time series is not a compact subset of a trajectory (intermediate states are hidden) and the iteration function is taken to include a 'secret parameter' - the 'key'. We can think of the sample as being 'random', 'unpredictable' and 'complex'. What do these properties mean mathematically and how do they relate to chaos? This paper focuses on answers to this question. In addition to probabilistic properties, we consider **algorithmic complexity**, i.e. the length of the shortest algorithm capable of producing a cryptographically secure sequence.

Intuitively, the internal complexity of a system provides its external unpredictability and a sequence is called **algorithmically random** if its algorithmic complexity equals the length of the sequence. An algorithmically random sequence is **computationally incompressible** and contains no recognizable patterns (redundancies). Clearly, a purely random system is also algorithmically random. However, the concepts of pseudo and algorithmic randomness are different; a pseudo-random string is generated with a compact seed, but the external observer is not able (practically) to reconstruct the generator and predict the sequence. In other words, the string is highly compressible for authorized communicators but computationally incompressible for the potential adversary and, in a general sense, an algorithmically random string can be predicted by a probabilistic machine.

Randomness or unpredictability can be 'measured' using such properties as algorithmic complexity and/or entropy, i.e. the degree of uncertainty about the system. Quantitatively, the Shannon entropy is in direct proportion to the algorithmic complexity in ergodic systems, where statistical properties of a single sequence coincides with that of all sequences generated by a PRNG. A randomness measure for chaos is the Kolmogorov-Sinai entropy that is, roughly speaking, a multi-resolution integration of Lyapunov exponents.

II. Complexity Theoretic Approach

In this paper we use a common terminology based on complexity theory [1] and, for completeness, we provide a brief introduction to the subject.

A. Turing Machine

A Turing machine is a hypothetical device that can theoretically implement any computer algorithm. It provides a unified framework to measure the complexity (i.e. program length and working time) of computational problems such as generating, transforming and matching cryptographic sequences. We denote a Turing machine as

$$T = \langle S, \mathcal{A}, \Gamma, F, q_0 \rangle \,,$$

where S is the finite state set of the control, $\mathcal{A}$ is the finite tape alphabet ($\mathcal{A} = \{0,1\}$), Γ is a finite rule state of the form $\gamma : S \times \mathcal{A} \to S \times \mathcal{A} \times \{L, N, R\}$, $F \subseteq S$ is a set of halting accepting states and $q_0 \in S$ is the initial state. The state $\{L, N, R\}$ includes 'tape commands' such as: 'move left' (L), 'stay in place' (N) and 'move right' (P). The *machine configuration* T is the triplet $\langle s, \alpha, i \rangle$ where $s \in S$ is the current state, $\alpha \in \mathcal{A}^*$ is the 'tape string' and $1 \leq i \leq |\alpha|$ is the head position counting from the left to the end of the tape. The machine is initialized in the following way: (i) a string $\alpha \subset \mathcal{A}^*$ is loaded onto the tape; (ii) the head is seeded to the leftmost position; (iii) the initial state q is assigned to the state variable s. At every step, (i) the machine reads a symbol from the current cell, depending on the symbol read and the current state, (ii) it makes a transition to a new state; (iii) it overwrites the current cell with a new symbol; (iv) it moves the head one step left or right, or stays in place. The machine halts when a state $f \in F$ is reached.

A Turing machine is said to *accept* a string α if a sequence of rules $\gamma_1, \ldots, \gamma_m \subset \Gamma^*$ exists that puts the machine from an initial state s_0 to any halting accepting state $f \in F$. The machine *rejects* a string, if it halts in $s \notin F$ or if it never halts. A *language* $\mathcal{L}$ over a finite alphabet $\mathcal{A}$ is a subset $(A)^\star$, i.e. a subset of all finite strings over $\mathcal{A}$. A machine is said to accept a language $\mathcal{L}$ if it accepts all the strings $\alpha \in \mathcal{L}$ and rejects $\beta \notin \mathcal{L}$. A *deterministic Turing machine* has single-valued rules $\gamma : S \times \mathcal{A} \to S \times \mathcal{A} \times \{L, N, R\}$, i.e. there are no rules with the same parts $S \times \mathcal{A}$. By contrast, a *non-deterministic machine* may have multi-valued rules. If there exists a polynomial $p\,(l)$ limiting the machines working time m (the number of steps) that depends of the input string length l ($m < p\,(l)$), the machine is said to run in polynomial time. The complexity class $\mathbf{P}$ is the set of languages accepted by a deterministic polynomial-time machines. The complexity class $\mathbf{NP}$ is the set of languages accepted by non-deterministic polynomial-time machines.

A *probabilistic Turing machine* is a deterministic Turing machine that can flip a fair coin to determine its next move. A probabilistic machine is said to accept a language $\mathcal{L}$ if it enters an accepting state for $\alpha \in \mathcal{L}$ with probability $p_1 > 2/3$ and halts $\alpha \notin \mathcal{L}$ with probability $p_0 > 2/3$. The complexity class $\mathbf{BPP}$ consists of all languages recognized by probabilistic polynomial-time machines.

B. Algorithmic Complexity

The concept of algorithmic complexity was suggested independently by three mathematicians A. N. Kolmogorov, P. Solomonov and G. J. Chaitin:

Definition 1: The algorithmic complexity $K_M\,(\alpha)$ of a finite string $\alpha \in \{0,1\}^n$ with respect to a Turing Machine M is the length $l(\pi)$ of the smallest computer program π, which generates it, i.e.

$$K_M\,(\alpha) = \min_{\pi : M(\pi) = \alpha} l(\pi).$$

Kolmogorov showed, that there exists a *universal* Turing machine U, that performs computations equivalent to π (designed for an arbitrary machine M) and that the changes in π required to adopt it for U depend on M but not on α. Consequently, the algorithmic complexity K_M with respect to any machine M is related to $K_U(S)$ by

$$K_U(S) \leq K_A(S) + C_A, \tag{1}$$

where C_M is a constant, which is independent from α. Here after, we omit the subscript U assuming that $K(\alpha) = K_U(\alpha)$. Unfortunately, algorithmic complexity cannot be commuted i.e. there is no universal solution for simplifying programs and for proving that the length is minimal. Thus, we cannot apply this definition directly to compare the complexity of cryptographic sequences or algorithms. Nevertheless the theoretical applications are very important. In particular, Kolmogorov complexity provides a unified approach to the problem of data compressibility.

C. Compressibility and Algorithmic Randomness

A string α_n of length n is said to be c-incompressible if $K(\alpha_n) \geq n - c$. Incompressible strings (where $c = 0$ or else is relatively small) are called algorithmically random.

D. Symbolic Complexity

For an infinite string α_∞ or a generator, it is interesting to consider the symbolic complexity given by the limit

$$c(\alpha_\infty) = \lim_{n \to \infty} \frac{K(\alpha_n)}{n}. \tag{2}$$

From (1) it follows that the symbolic complexity $c(\alpha_\infty)$ is invariant with respect to the choice of Turing machine. If a string has a finite Kolmogorov complexity (e. g. a pseudo-random string), its symbolic complexity tends to 0. A truly random string has $c = 1$ because its length equals the length of the shortest program. Clearly, $c > 0$ if and only if the generator has infinite complexity. In chaotic systems, this happens if the complexity of the initial conditions is infinitely large or a certain amount of randomness is introduced into the system from the environment.

III. Information Theoretic Approach

In ideal cryptosystems, the distribution of the ciphertext cannot be differentiated from uniform noise and thus provides no useful information for an adversary.

A. True Randomness

We define a Probability Distribution Function (PDF) as a function from strings $\mathcal{L} = \{\alpha_j\}$ to nonnegative real numbers, i.e. $\Pr : \mathcal{L} \rightarrow [0,1]$ such that $\sum\limits_{\alpha \in \mathcal{L}} \Pr(\alpha) = 1$.

Definition 2: A string α is called truly (purely) random (or unpredictable) if, for any substrings, $\beta_n, \gamma_n \in \alpha$, $0 > n > length(\alpha)$

$$\Pr(\beta_n) = \Pr(\gamma_n)$$

.

A truly random string cannot be predicted, i.e. for any symbol $s_i \in \alpha$, the conditional probability $\Pr(s_i | s_{i-1}, s_{i-2}, \ldots) = \Pr(s_i)$. In other words, an arbitrary large amount of knowledge about the previous states does not increase the probability of the successful prediction of the next state. An infinite and truly random string has a delta autocorrelation function and an infinite and uniform power spectrum (white noise).

B. Shannon Entropy

The Shannon entropy measures the amount of information required to determine precisely the system state among all possible states [2]. In cryptography, the entropy is related to the unpredictability of an encryption system for an adversary. The entropy of a string α_n of length n is defined as

$$H_n = -\sum_{\alpha \in \mathcal{A}^n} \Pr(\alpha_n) \log_{|\mathcal{A}|} \Pr(\alpha_n), \tag{3}$$

where $\Pr : \mathcal{A}^n \rightarrow [0,1]$ is the PDF of α_n on the set of n^{th} symbol strings. The maximum of H_n is achieved when $\Pr(\alpha_n)$ is a uniform distribution and the string is truly random. The conditional entropy h_n denotes the average amount of information supplied with each $(n+1)^{\text{th}}$ symbol provided the previous n symbols are known:

$$h_n = h_{n+1|n} = \begin{cases} H_{n+1} - H_n, & n \geq 1 \\ H_1, & n = 1 \end{cases}$$

In other words, h_n quantifies the average uncertainty when predicting the next symbol. As soon as knowledge about a previous state cannot increase the uncertainty, the function H_n is non decreasing and $h_{n+1} \leq h_n$. For a stationary information source there exists a limit

$$h_{Sh} = \lim_{n \to \infty} h_n = \lim_{n \to \infty} \frac{H_n}{n}, \tag{4}$$

called the entropy of information source (cryptographic system). Further, if α is a k^{th} order Markov sequence, then $h_n = h_{Sh}$ for all $n \geq k$. A Markov sequence corresponds to a deterministic process, in which the next state depends on the previous k states, i.e. for $\quad s_i \in \alpha$

$$\Pr(s_i | s_{i-1}, s_{i-2}, \ldots) = \Pr(s_i | s_{i-1}, s_{i-2}, \ldots, s_{i-k})$$

Examples of Markov processes can be found in most cryptographic systems such as PRNG's and block ciphers.

C. Entropy-Complexity Relationship

Intuitively, complexity and the entropy are related in terms of 'cause and effect': the more complex the internal organization of a system, the more unpredictable its behavior is and the higher the entropy becomes. The complexity is the size of the 'internal program' that generates a state sequence (string), whereas the entropy is computed from the probability distribution of this sequence. Formally, the following result can be applied for stationary ergodic sources [3]:

$$\lim_{n \to \infty} \frac{\langle K_n \rangle}{H_n} = \frac{1}{\ln 2}, \tag{5}$$

where

$$\langle K_n \rangle = \sum_{\alpha_n \in \{0,1\}^n} \Pr(\alpha_n) K(\alpha_n)$$

Hence, the average complexity $\langle K_n \rangle$ is asymptotically proportional (with the coefficient $\ln 2$) to the entropy as n increases.

IV. Entropy and Complexity

A. Partitioning and Symbolic Dynamics

Consider a chaotic system $\mathcal{S} = \langle X, f \rangle$ with an f-invariant measure μ. Any set of m disjoint regions that covers the state space X is a partition denoted by

$$\beta = \{X_1, X_2, ..., X_m\} :$$

$$\bigcup_{i=1}^{i=m} X_i = X, \quad X_i \cap X_j = \emptyset, \; \forall i \neq j.$$

A unique symbol $s_i \in \mathcal{A}$ is assigned to every region X_i. The process of partitioning the state space and assigning symbols to every region from the partition resulting in macroscopic dynamics is called *symbolic dynamics* [4]. A function σ defines partitions and their symbolic names as follows:

$$\sigma(x) = \{s_i \in \mathcal{A} | x \in X_i\}.$$

A trajectory $\phi(x_0)$ passing across the subsets X_i produces a *symbolic trajectory* $\alpha(x_0)$.

B. Kolmogorov-Sinai Entropy

The Lyapunov exponents measure how fast we lose the capability to predict the behavior of a chaotic system in time. The disadvantage is that this measure does not consider the resolution under which the system is observed, unlike the Kolmogorov-Sinai entropy [3]. Let the partition $\beta = \{X_1, X_2, ..., X_m\}$ be the observer's resolution. Looking at the system state x, the observer can only determine the fact that $x \in X_i$ and reconstruct the symbolic trajectory $\alpha_n = \{s_{m_1}, s_{m_2}, \ldots, s_{m_n}\}$ corresponding to the regions visited. The entropy of a trajectory α_n with respect to partition β is given by

$$H_n^{\beta} = -\sum_{\alpha_n} \Pr(\alpha_n) \log_{|\mathcal{A}|} \Pr(\alpha_n),$$

where $\Pr(\alpha_n)$ is the probability of occurrence of the substring α_n. The conditional entropy of the $(n+1)^{\mathrm{th}}$ symbol provided the previous n symbols are known is defined as

$$h_n^{\beta} = h_{n+1|n}^{\beta} = \begin{cases} H_{n+1}^{\beta} - H_n^{\beta}, & n \geq 1 \\ H_1^{\beta}, & n = 1 \end{cases}$$

The entropy for a partition β is given by

$$h^{\beta} = \lim_{n\to\infty} h_n^{\beta} = \lim_{n\to\infty} \frac{1}{n} H_n^{\beta}.$$

The Kolmogorov-Sinai entropy of a chaotic system is the supremum over all possible partitions

$$h_{KS} = \sup_{\beta} h^{\beta}. \qquad (6)$$

The KS entropy equals zero for regular systems, is finite and positive for deterministic chaos and infinite for a truely random process. It is related to the Lyapunov exponents by $h_{KS} = \sum_{1\leq d\leq D} \lambda_d$ and proportional to the time horizon T on which the system is predictable.

C. Complexity of a Trajectory

The complexity of a trajectory at a point x_0 with respect to a finite open coverage β is defined as

$$\mathcal{C}^{\beta}(x_0) = \limsup_{n\to\infty} \frac{1}{n} \min_{\alpha_n \in [\psi(x)]^n} K(\alpha_n),$$

where $[\psi(x)]^n = \left\{\alpha_n | f^j(x_0) \in X_j\right\}$ and $K(\alpha_n)$ is the algorithmic complexity of α. The complexity of the trajectory of a point x_0 is

$$\mathcal{C}(x_0) = \sup_{\beta} \mathcal{C}^{\beta}(x_0).$$

Definition 3: (algorithmically random trajectory, [5], [6]) The trajectory of a point x_0 is called algorithmically random if its complexity is positive, i.e. $c(x_0) > 0$.

The Brudno-White theorem defines the relationship between the KS entropy and complexity:

Theorem 1: (complexity of the trajectory, [5], [6]) The symbolic trajectories of almost all $x \in X$ (with respect to the invariant measure μ) are algorithmically random and their complexity is given by

$$c(x) = \frac{h_{KS}}{\ln 2}, \qquad (7)$$

Though it is practically impossible to quantify the algorithmic complexity of a string, most strings over a finite alphabet produced by a chaotic system and are algorithmically random.

V. Pseudo-Randomness

A. Probabilistic Ensembles

Let $\Pr_i(\alpha)$ be a probability distribution function of strings $\{0,1\}^{l(i)}$, where $l(i)$ is a positive polynomial. We write $\Pi = \{\Pr_i\}_{i\in I}$ for an ensemble of distributions indexed by $I \subset \mathbb{N}$. The ensemble of the uniform distributions $\Pi_0 = \{\Pr_{0,i}\}_{i\in\mathbb{N}}$ for all $i \in \mathbb{N}$ and $\alpha, \beta \in \{0,1\}^i$ satisfies $\Pr_{0,i}(\alpha) = \Pr_{0,i}(\beta)$. To measure the 'degree of randomness' of a string, its probability ensemble should be compared with that of the uniform distributions. Having limited resources, computers can process only a subset of distributions. Thus, we introduce the concept of polynomial indistinguishability. Roughly speaking, two probabilistic ensembles are polynomially indistinguishable if they assign 'about the same' mass to the same subsets of strings, efficiently recognized by a Turing machine:

Definition 4: (polynomial indistinguishability, [7], [8], [9]) Let $\Pi_1 = \{\Pr_{1,i}\}_{i\in I}$ and $\Pi_2 = \{\Pr_{2,i}\}_{i\in I}$ be two probability ensembles each indexed by I. Let T be a probabilistic polynomial-time Turing machine called a test. The test gets two inputs: an index i and a string α. Let $\Pr_1^T(i)$ be the probability that, on input index i and a string α chosen according to the distribution $\Pr_{1,i}$, the test T outputs 1. Similarly, $\Pr_2^T(i)$ denotes the probability that, on input index i and a string α chosen according to the distribution $\Pr_{2,i}$, the test T outputs 1. We say that Π_1 and Π_2 are *indistinguishable* with polynomial $p(i)$ if for all probabilistic polynomial-time tests T and sufficiently large $i \in \mathbb{N}$

$$\left|\Pr_1^T(i) - \Pr_2^T(i)\right| < \frac{1}{p(i)}.$$

Definition 5: (pseudo-random probability ensemble, [7], [8], [9]) The probability ensemble $\Pi = \{\Pr_i\}_{i\in I}$ is said to be pseudo-random if, for any positive polynomial $p(i)$, the ensemble Π is indistinguishable from $p(i)$ with uniform ensemble $\Pi_0 = \{\Pr_{0,i}\}_{i\in I}$.

Definition 6: (unpredictable probability ensemble, [7], [8], [9]) Let $\Pi = \{\Pr_{1,i}\}_{i\in I}$ be a probabilistic ensemble indexed by I. Let T be a probabilistic ensemble polynomial-time Turing machine that on input (index i and a string α), outputs a single bit, called the guess. Let $bit(\alpha, r)$ denote the r^{th} bit of the sequence α and $pref(\alpha, r)$ denote the prefix of r bits of the string α, i.e. $pref(\alpha, r) = bit(\alpha, 1)\, bit(\alpha, 2) \ldots bit(\alpha, r)$. We say that the machine T predicts the next bit of Π, if for some polynomial $p(i)$ and infinitely many i's,

$$\Pr\left(M\left(i, pref(\alpha, r)\right) = bit(\alpha, r+1)\right) \geq \frac{1}{2} + \frac{1}{p(i)},$$

where the probability space is that of the string α chosen according to $\Pr_{1,i}$ and the integer r is chosen at random with uniform distribution in $\{0, 1, \ldots, l(\alpha) - 1\}$. We say that Π is unpredictable if there exists no probabilistic polynomial time machine T which predicts the next bit of Π.

Theorem 2: [8], [9], [10] The probability ensemble Π is pseudo-random if and only if Π is unpredictable.

B. One-Way Functions

One-way functions are functions that are easy to evaluate ($\beta = f(\alpha)$), but hard (on average) to invert ($\alpha = f^{-1}(\beta)$) and lie at the heart of modern cryptography, in particular, their use in public-key schemes. The computational gap between forward and inverse evaluation quantifies the efficiency of the one-way transformation. A formal definition of a one-way function is given in terms of complexity theory:

Definition 7: (one-way function [10], [8], [9]) A function $f : \{0,1\}^* \to \{0,1\}^*$ is called one way if it satisfies the following: (i) there is a deterministic polynomial-time Turing machine that on input α returns $f(\alpha)$; (ii) for any probabilistic polynomial-time Turing machine M, any positive polynomial $p(n)$ and sufficiently large n

$$\Pr(M(f(\alpha), 1^n) \in f^-1(\alpha)) < \frac{1}{p(n)},$$

where the probability is taken over all possible choices of $\alpha \in \{0,1\}^n$ and the internal tosses of M conform to a uniform probability distribution. The role of 1^n is to allow machine M to run in a time polynomial over the length of the pre-image it is supposed to find.

A stronger notion of unpredictability is that of a hard-core predicate. A polynomial-time computable predicate b is called a hard-core of a function f if all algorithms, given $f(\alpha)$, can guess $b(\alpha)$ only with a probable success which is negligibly better than a half.

Definition 8: (hard-core predicate, [10], [8], [9]) Let $f : \{0,1\}^* \to \{0,1\}^*$ and $f : \{0,1\}^* \to \{0,1\}$. The predicate b is said to be hard-core of the function f, if: (i) there is a deterministic polynomial-time Turing machine that on input α returns $b(\alpha)$; (ii) there is no probabilistic polynomial-time Turing machine M such that for any positive polynomial $p(n)$ and sufficiently large n

$$\Pr(M(f(\alpha), 1^n) = b(\alpha))) < \frac{1}{2} + \frac{1}{p(n)},$$

where the probability is taken over all possible choices of $\alpha \in \{0,1\}^n$ and the internal tosses of M conform to a uniform probability distribution.

Theorem 3: (existence of a one-way function with a hard-core predicate, [7], [8]) If there exists a one-way function, then there exists a one-way function with a hard-core predicate.

C. Pseudo-Random Number Generators (PRNGs)

In general, a PRNG is an efficient (deterministic) algorithm that on input of a short seed (initial condition), outputs a typically much longer sequence that is computationally indistinguishable from a uniformly chosen string.

Definition 9: (pseudo-random generator, [10], [8]) Let $l : \mathbb{N} \to \mathbb{N}$ satisfy $l(n) > n$ for all $n \in \mathbb{N}$. A pseudo-random generator, with a stretch function $l(n)$, is a deterministic polynomial time algorithm G satisfying the following:

1) For every $\alpha \in \{0,1\}^*$ it holds that $|G(\alpha)| = l(|\alpha|)$
2) The probabilistic ensembles $\Pi = G(\Pr_0^n)$ and $\Pi_0^{p(n)}$ are computationally undistinguishable.

Theorem 4: (construction of a pseudo-random generator, [7], [8]) Let f be a one-way $1:1$ function and b be a hard-core predicate of f. Then

$$G(\alpha) = b(\alpha)b(f(\alpha)) \ldots b(f^{l(|\alpha|)-1}(\alpha))$$

is a pseudo-random generator with a stretch function l.

Consequently, a pseudo-random generator can be constructed from any one-way length-preserving function (rather then merely one-way permutations). On the other hand, the existence of a one-way function is a necessary condition to the existence of the pseudo-random generator, that is

Theorem 5: (existence of pseudo-random generators, [8]) Pseudo-random generators exist if and only if one-way functions exists.

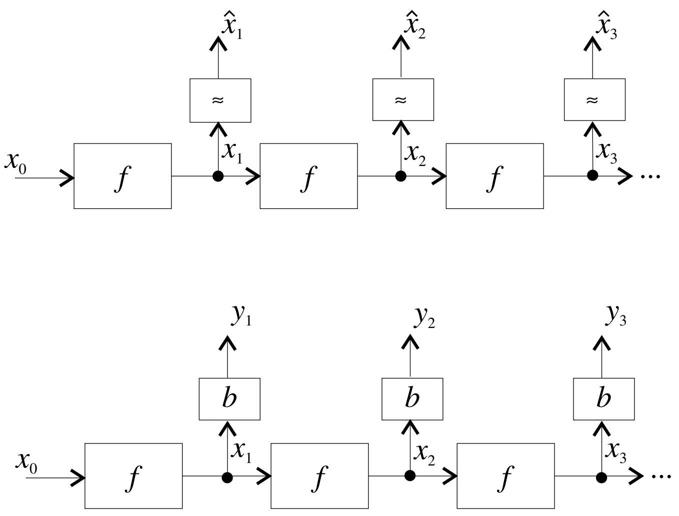

Fig. 1. A synergy between a chaotic system (top: $\approx$ is a rounding function, $\hat{x_n}$ is the output) and a PRNG (bottom: b is a hard-core predicate, y_n is the output).

Assuming the existence of one-way 1:1 functions, there can exist probability distributions that are non-uniform and are not even statistically close to being uniform but are nevertheless computationally indistinguishable from a uniform distribution [9]. The definition of a pseudo-random generator given above cannot be applied directly since there is no practical way to prove or check rigourously indistinguishability.

VI. Discussion

Practical cryptography is based on passing known statistical tests [17], which ensure the pseudo-random property of a generator. Moreover, it is considered that pseudo-random sequences can be used instead of truly random sequences in most cryptographic applications. Although there is a synergy between pseudo-random generators and chaotic systems there is a also fundamental and important difference which is that the iterated function of a chaotic system is not required to be one-way. Chaos theory pays no attention to the algorithmic complexity of f and f^{-1}, which is one of the main problems associated with the applications of chaos theory to cryptography. However, based on the study provided, we now present the design methods and example algorithms required to implement chaos to encrypt data.

VII. Applications of Chaos for Digital Cryptography

From a theoretical point of view, chaotic systems produce infinite random strings that are asymptotically uncorrelated. This property relates to genuine chaotic systems with an infinite number of states. For applications to digital cryptography, a finite-state systems approach is required which puts certain constraints on the design of the algorithm(s). In this paper, we study these constraints and present the principal criteria required to design meta-encryption engines using pseudo-chaotic algorithms.

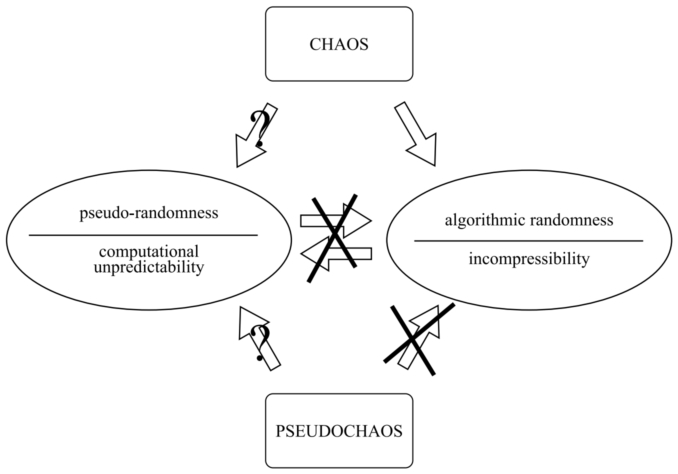

Fig. 2. Properties of chaotic and pseudo-chaotic systems.

The notion of *pseudo-chaos* introduced in [18], 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; (ii) 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 in Part I 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? 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.

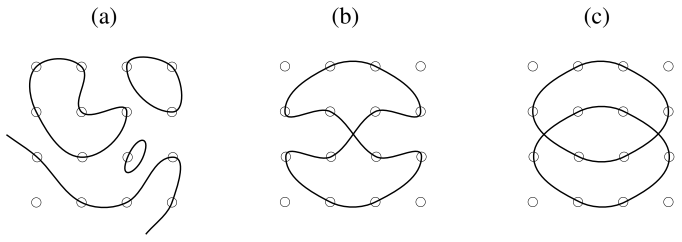

Fig. 3. Examples of orbits of a pseudo-chaotic system. (a) Dangerously short orbits (unsuitable for cryptography); (b) A single orbit (the best choice for cryptography); (c) Multiple orbits with the same length (also suitable for encryption).

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 [19]. 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.

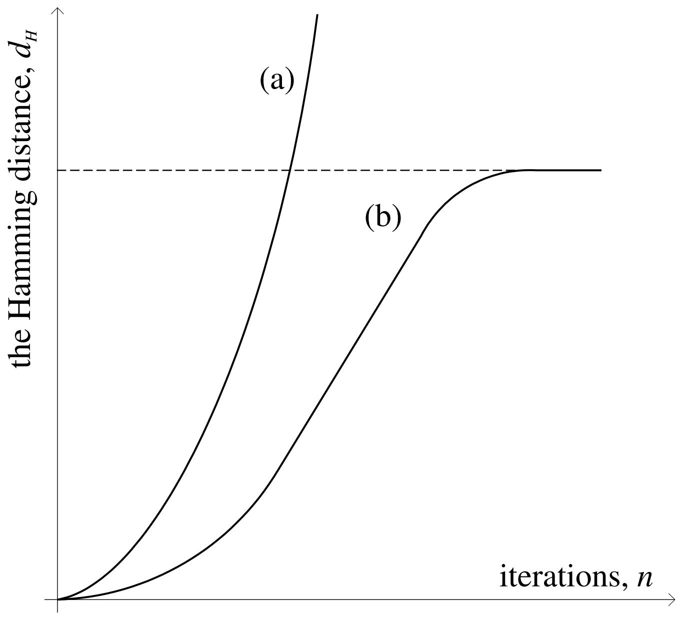

Fig. 4. A Laypunov exponent of a chaotic (a) and pseudo-chaotic (b) system

A comparison between the average Lyapunov curve of a chaotic system and an analogous pseudo-chaotic system is given in Figure 4. If the pseudo-chaotic system has a finite precision σ, then the exponential divergence given by

$$e^{n\lambda} = \frac{\left| f^n (x_0 + \varepsilon) - f^n (x_0) \right|}{\varepsilon}, \qquad n \to \infty, \quad \varepsilon \to 0, \quad (8)$$

will eventually be limited by $\varepsilon = \sigma$. Usually the fraction (8) grows exponentially during the first few iterations and then increases linearly until it finally levels off at a certain finite value.

VIII. Floating-point Approximations

Floating-point and fixed point arithmetic are the most straightforward solutions for approximating a continuous system on a finite state machine [20]. 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

$$round_k(x) =$$

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

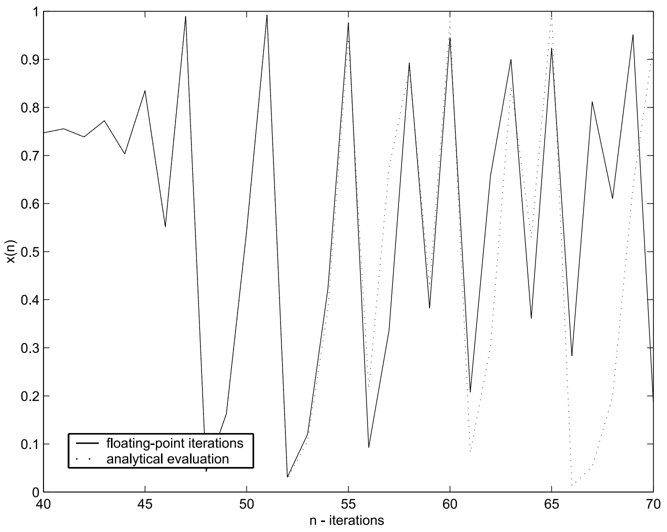

Fig. 5. Trajectories of a continuous-state chaotic system (9) and its 64-bit floating-point approximation. The first curve is obtained by means of the analytical solution (10). The rounding off error is amplified at each iteration and the trajectories diverge exponentially.

The iterative rounding is accumulative and results in surprisingly different behavior of pseudo-chaos compared with the continuum counterpart. Figure 5 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

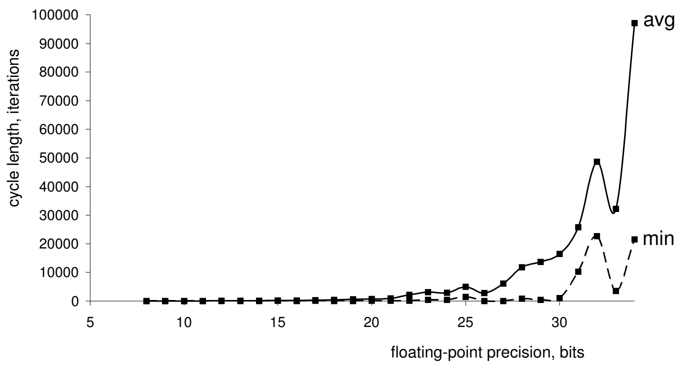

Fig. 6. The average and the minimal cycle length of the logistic system (9) verses floating-point precision obtained from 10 samples of the logistic system.

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 6 is a plot of the average cycle length verses floating-point precision and shows that high precision does not guarantee a sufficiently long trajectory.

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 (as discussed in Part I) to digital cryptography and in the following sections, an overview of encryption schemes based on a floating-point approximation to chaos is given.

IX. 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 9 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.

X. Example Chaotic Map

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

$$x_{n+1} = 4rx_n\left(1 - x_n\right), \quad (9)$$

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

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

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

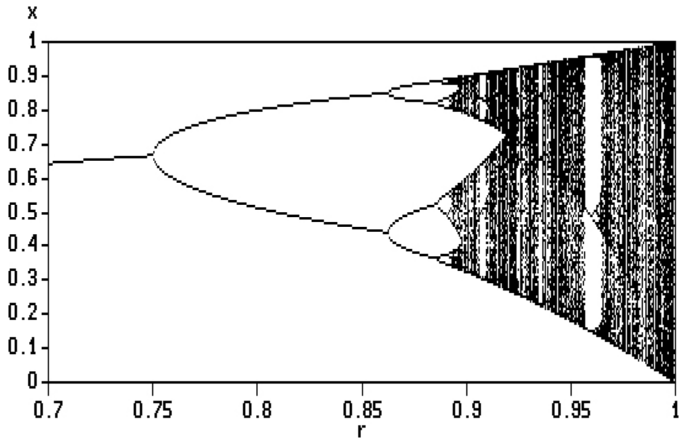

Fig. 7. Bifurcation of the logistic map. The most 'unpredictable' behavior occurs when $r \rightarrow 1$.

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

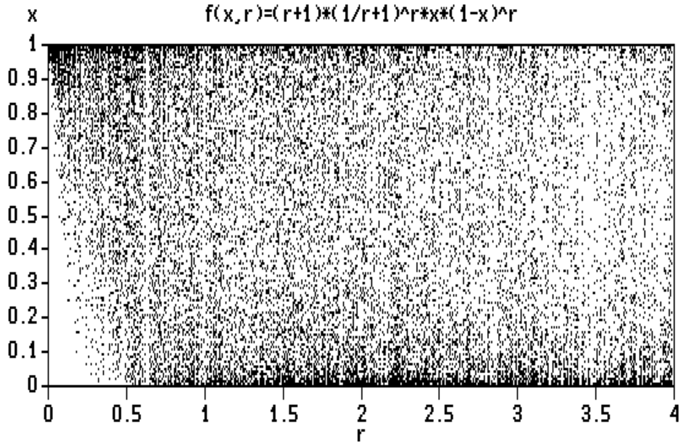

Fig. 8. Attractor points corresponding to different values of the parameter r in the Matthews map.

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 [21], [22] is:

$$x_n = \sin^2\left(2^n \arcsin\sqrt{x_0}\right). \quad (10)$$

When $n = 1$ we have the initial equation (9). Hence, the state x_n can be computed directly from x_0 without performing n iterations.

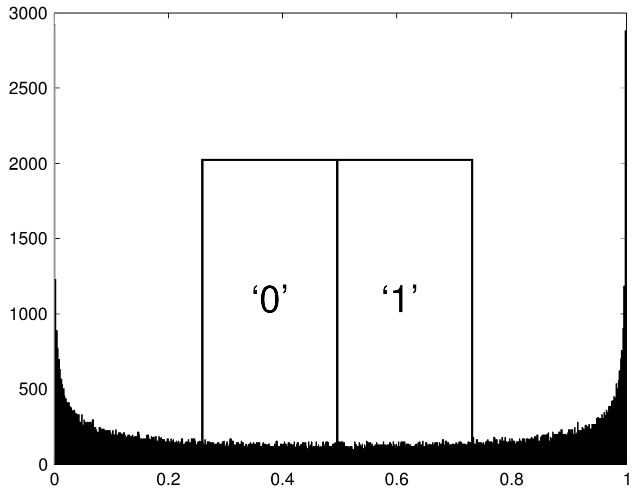

Fig. 9. The Probability Density Function of a state sequence produced by the logistic system with an incomplete partition.

Bianco *et al.* [23] used the logistic map (9) 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 9). 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 [24] and Jackson [25] 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 6).

B. Matthews Map

Matthews [26] 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.

C. Other Examples of Chaotic Maps

Gallagher *et el.* [27] 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 [28] 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 [29]. 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 [30] 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

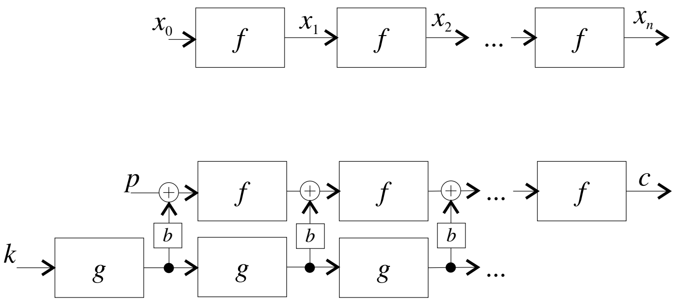

Fig. 10. A typical block cipher is a combination of several pseudo-chaotic systems

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 [31]. 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 10).

F. Multi-Algorithmic Generators

Protopopescu [32] 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: (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. This can be 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

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

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

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

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

XI. Conclusion

A demonstration encryption system based on multiple chaotic systems with extended properties (i)-(iv) is available from http://eleceng.dit.ie/arg/downloads/crypstic The system solves the problems related to the floating-point arithmetic in a 'extensive' way to provide $(m - 1)$ redundant systems. It is noted that this system 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 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 [33]: *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 programme.

References

[1] L. Lovász. Computation complexity. Lecture Notes. http://ftp.cs.yale.edu/pub/lovasz.pub/.

[2] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(4):379–423, 623–526, 1948.

[3] G. Boffetta, M. Cencini, M. Falcioni, and A. Vulpiani. Predictability: a way to characterize complexity, 2001. http://www.unifr.ch/econophysics/.

[4] B.-L. Hao. *Elementary symbolic dynamics and chaos in dissipative systems*. World Scientific Pub Co, 1989.

[5] A. A. Brudno. Entropy and the complexity of the trajectories of a dynamical system. *Trans. Moscow Mathematical Society*, 44, 1983.

[6] H. White. Algorithmic complexity of points in dynamical systems. *Ergodicity Theory Dynamical Systems*, 13, 1993.

[7] A. C. Yao. Theory of applications of trapdoor functions. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

[8] O. Goldreich. Introduction to complexity theory. Lecture Note, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel, 1999.

[9] L. Kocarev. Chaos and cryptography, 2001. http://rfic.ucsd.edu/chaos/ws2001/kocarev.pdf.

[10] M. Blum and S. Micali. How to generate crytographically strong sequences of pseudo-random bits. *SIAM Journal of Computation*, 13(4):850–864, 1984.

[11] L. A. Levin. One-way function and pseudorandom generators. *Combinatorica*, 7(5):357–363, 1987.

[12] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proc. 21st Annu. ACM Symp. on Theory of Computing*, pages 230–235, 1989.

[13] J. Hastad. Pseudo-random generators under uniform assumptions. In *Proceedings 22nd Annu. ACM Symp. on Theory of Computing*, pages 385–404, 1990.

[14] T. Ritter. Ciphers by ritter, 1991. http://www.ciphersbyritter.com/.

[15] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. NIST, 2001. http://csrc.nist.gov/rng/rng2.html.

[16] George Marsaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995. http://stat.fsu.edu/pub/diehard/.

[17] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptology*. CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac/.

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

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

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

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

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

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

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

[25] E. A. Jackson. Perspectives in nonlinear dynamics. Cambridge University Press, 1991.
[26] R. Matthews. On the derivation of a chaotic encryption algorithm. *Cryptologia*, (13):29–42, 1989.
[27] J. B. Gallagher and J. Goldstein. Sensitive dependence cryptography, 1996. http://www.navigo.com/sdc/.
[28] 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.
[29] N. Paar. Robust encryption of data by using nonlinear systems, 1999. http://www.physik.tu-muenchen.de/~npaar/encript.html.
[30] T. Ritter. The efficient generation of cryptographic confusion sequences. *Cryptologia*, (15):81–139, 1991. http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM.
[31] V. Rijmen and J. Daemen. Rijndael algorithm specification, 1999. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.
[32] V. A. Protopopescu, R. T. Santoro, and J. S. Tolliver. Fast and secure encryption-decryption method. US Patent No. 5479513, 1995.
[33] O. Hoare, Enigma: Codebreaking and the Second World War. The True Story through Contemporary Documents, introduced and selected by Oliver Hoare. UK Public Record Office, Richmond, Surrey, 2002.