ORNL/TM-12440

Cryptosystems Based on Chaotic Dynamics

R. A. McNees
V. Protopopescu
R.T. Santoro
Engineering Physics and Mathematics Division

J. S. Tolliver
Computing Applications Division

DATE PUBLISHED - AUGUST 1993

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

TABLE OF CONTENT

ABSTRACT

An encryption scheme based on chaotic dynamics is presented. This scheme makes use of the efficient and reproducible generation of cryptographically secure pseudo random numbers from chaotic maps. The result is a system which encrypts quickly and possesses a large keyspace, even in small precision implementations. This system offers an excellent solution to several problems including the dissemination of key material, over the air rekeying, and other situations requiring the secure management of information.

1. INTRODUCTION

In this paper, we investigate the potential for employing chaotic dynamics as a reliable, secure, and rapid encryption tool in symmetric cryptography. The basic idea of cryptography is to alter a message in such a way as to make it unintelligible to anyone except the intended recipient.[1,2] The original message, M, also referred to as the plaintext, is represented by a finite string of symbols from a given alphabet, S_m. The encryption procedure codes the message using a transformation, E, that depends on a set of parameters, K, called the key. The result is an encrypted ciphertext

$$C = E(M;K)$$

which is meaningless to an unintended observer. In a symmetric cryptosystem, the recipient of the ciphertext retrieves the original message by using the same key as the sender and employing a decryption transformation, D:

$$D(E(M;K);K) = M.$$

Both the sender and recipient of the message must share the same key for the message to be successfully interpreted.

In the past, cryptography was used primarily within the military, intelligence, and diplomatic communities. With the increased speed and facility of data transfer allowed by modern computer systems, cryptographic applications have also appeared in banking, personnel administration, computer networking, and counter-narcotics activities.

The emergence of new cryptographic concerns in non-traditional areas of interest has led to the development of several *iterated cryptosystems*. An iterated cryptosystem relies upon the repeated application of weak functions to produce cryptographically strong results. The most popular of these, the Data Encryption Standard (DES), was adopted by the National Bureau of Standards in 1977.[3] The DES, along with a large number of cryptosystems inspired by it, survived attempts at attack for several years. However, in recent years effective attacks on these systems have appeared, based on the method of *differential cryptanalysis*. This method has exposed design flaws not only in DES, but in many other iterated cryptosystems, showing that the time required to defeat some of these schemes can in many cases be reduced to a matter of minutes or even seconds on personal computers[4]. Although DES itself appears relatively secure at this time, the fact that it has revealed exploitable weaknesses increases the need for alternative cryptosystems.

A drawback of iterated cryptosystems is the extreme difficulty associated with *proving* their security. One way to avoid this problem is to develop a cryptosystem working from an a priori strong foundation. An example of such a system is the *one-time pad* cryptosystem. The one-time pad is the only cryptosystem which can be proven to be secure. However, in light of the severe requirements it demands in order to guarantee security, the one-time pad is not practical for general use.

The one time pad requires, for any plaintext message M composed of i bits, a unique and random string K, also consisting of i bits, with a uniform distribution. The space of all possible strings K is the *keyspace*. Encryption of the plaintext is achieved through simple combination of these two strings by some bitwise mechanism, in this case we will define the ciphertext C to be the exclusive-or (XOR) product of M and K. The distribution of the random string K is uniform and independent of the distribution of M, which implies that the distribution of C is uniform and independent of the distribution of M as well. Since the string K is random, any attempt to decrypt the ciphertext, without knowledge of the string K, possesses only a minimal chance of success.

As mentioned above, the proper use of the one time pad entails requirements which greatly limit its practicality. The first requirement is obvious; the one time pad requires the secure distribution of as much key material as plaintext. Second, a new random string must be used for each encryption, as attacks employing multiple ciphertexts encrypted under the same key are trivial.[4] The impracticalities associated with these two requirements are referred to as the *key management* problem. Making use of the one time pad as an effective foundation for a new cryptosystem requires the elimination of the key management problem. In order to do this the amount of information needed to drive the cryptosystem must be significantly decreased without diminishing the scheme's security.

In the work reported here, we present a variation of the one time pad that yields a practical solution to the key management problem. The pseudo random behavior of chaotic dynamics is used to produce pseudo random sequences from a small amount of initial information. These pseudo random numbers are then combined with the plaintext message to generate the cipher text. This concept is supported by two properties of chaotic dynamics: [5,6] (i) its highly irregular character successfully mimics truly stochastic behavior and (ii) its deterministic nature ensures simple, rapid, and accurate reproducibility. A general discussion of cryptographically suitable methods of pseudo random number generation is presented in Section 2. The application of chaotic dynamics to a practical solution of the key management problem is discussed in Section 3. The actual implementation of chaotic dynamics in an encryption scheme is examined in Section 4, and preliminary conclusions derived from this work are summarized in Section 5.

2. PSEUDO RANDOM NUMBER GENERATION

The development of a secure cryptosystem invariably requires effective and secure random number generation. Some of the more popular random number generators in use today are based on the linear congruential method, the middle square method, multiplicative methods, and mixed methods. [7-8] These are enhanced by additional techniques such as data perturbation, swapping random sample queries, cell suppression, partitioning, and complex bitwise manipulation. These methods have met with varying degrees of success in different applications, but they do not provide a definitive answer to the random number generation problem.[9-10]

An ideal generator would produce a truly random sequence. However, this is impossible since both the generation and the analysis of a truly random sequence would require infinite information content. [11] An actual generator can, therefore, produce only a *pseudo random* sequence for which various measures of randomness can be defined.[7,8,12] For practical use in a given application, we require that a pseudo random number (PRN) generator possess: (i) reproducibility, (ii) computational efficiency, and (iii) adherence to standards related to that specific application.

For instance, consider the computational efficiency of a PRN generator. The generator must be both rapid in the production of a pseudo random sequence and economical in its storage. In some cases, there is a direct trade-off between the two qualities. A routine designed to generate numbers to be used to dynamically encrypt real time transfer of data is more concerned with the speed at which it can generate a pseudo random sequence. A routine intended to generate PRNs to be used in the encryption of electronic documents which are then stored must incorporate efficient storage considerations. The configuration which possesses the maximum utility for a particular application must therefore be determined based on the requirements of that application.

When employed within cryptographic applications, the PRN generator comes under the scrutiny of a well informed *enemy*, equipped with modern computational resources. The enemy's goal is to reproduce a particular sequence of pseudo random values. The enemy does not possess the unique initial information (i.e. initial values, seeds, and other variable parameters) associated with the sequence he wishes to regenerate. For the generator to be useful cryptographically, any attempt by the enemy to reproduce subsequent portions of a pseudo random sequence given a finite portion of that sequence (referred to as an *attack*) must have a trivial chance of success in any useful amount of time.

To insure security against the enemy, we avoid a purely statistical notion of randomness[2,7,8] and instead adopt a more cryptographically practical definition. Any statistical benefits incurred from a particular PRN generator which are not directly associated with its adherence to our cryptographic definition of random are cosmetic, and add little to the generator's usefulness. A cryptographically strong pseudo random number generator (CSPRNG) must produce sequences of values which: (i) possess *minimal internal correlation*, (ii) convey *minimal critical information* regarding their origin, and (iii) are absolutely dependent upon *unique* and *sensitive* initial conditions for proper reproduction.

In order to be useful cryptographically, a PRN generator must produce sequences with *minimal internal correlation*. By this, we mean a sequence of PRNs must possess an acceptably small correlation between subsequent values and close neighbors. Furthermore, any long range correlations (periodicity) are equally undesirable. The existence of such correlations can offer information regarding the nature of the CSPRNG used to produce the sequence. The availability of such information is contrary to the purpose of the generator and must be avoided.

The *critical information content* of the sequences generated by a CSPRNG must be carefully monitored. Critical information content is the quality of a sequence that associates it with the composition of a particular PRN generator and the specific parameters it employs. Output which completely retains critical information may be easily attributed to a particular PRN generator. Similarly, an output which retains minimal critical information can not practically be associated with any one particular method of PRN generation. For example, any member of an unaltered sequence of iterates resulting from some recursive process retains all the critical information necessary to recreate that sequence in either direction. In this sense, the critical information content of a sequence is directly related to the degree of internal correlation between its members. One method of visualizing the critical information content of a sequence is through the use of Poincaré plots, which display a member of a sequence, x_{n+i} ,versus another member, x_n. Depending upon the underlying dynamics of the PRN generator and the value of the lag i, such a plot eventually reveals a structure which is directly dependent upon the critical information content of the sequence.

A CSPRNG must require *unique* initial conditions for the generation of a pseudo random sequence, and be *sensitive* to any changes in those conditions. Ideally, each initial condition should eventually yield a unique pseudo random sequence, and no correlation should exist between two initial values and the similarity of the output they generate. In a realistic application, however, we do not exclude the possibility of multiple initial conditions resulting in the same output. This is acceptable as long as the number of such initial conditions is relatively small.

Based on these three requirements, using nonlinear maps in a chaotic regime to implement CSPRNGs appears promising. The reproducibility of sequences generated by these maps is guaranteed by their deterministic character. The computational efficiency of the generator is a result of their recursive nature. A computer-based application performs few operations per iteration, making the generation of long strings of iterates simple and quick. The sensitivity of chaotic maps to minute changes in initial condition insures that the generator will also be sensitive to such changes. Furthermore, statistical tests show that the output of chaotic maps can be efficiently transformed so as to relate minimal critical information and possess practically no internal correlation.

3. APPLICATION TO THE KEY MANAGEMENT PROBLEM

We present the following application of a chaotic dynamics-based CSPRNG to the key management problem.

The PRN generator performs all operations in b bit floating point precision. The first stage of the generator employs m nonlinear chaotic maps, C_1, C_2, ... C_m, operating on the unit segment. These maps require m initial values (seeds), each a b bit floating point value designated K_1, K_2, ... K_m. Each map iterates its respective seed l times, producing the iterates $C_1^{[l]}(K_1)$, $C_2^{[l]}(K_2)$, ... , $C_m^{[l]}(K_m)$. These iterates are combined via the *exclusive-or* operation, referred to as XOR and denoted by $\otimes$, to give a value R_l:

$$R_l = C_1^{[l]}(K_1) \otimes \ldots \otimes C_m^{[l]}(K_m) .$$

This number is then transformed into a pseudo random integer, $p_l = I(R_l)$, by extracting *one byte* (8 bits) from a specific address in the binary representation of R_l and expressing it as an integer. The value p_l is the first member of the pseudo random sequence of integers p_1, p_2, ..., p_n. Additional values are generated by the same process using the n^{th} set of chaotic iterates $C_i^{[nl]}(K_i)$, $i=1,...,m$, as the new keys for the $(n+1)^{th}$ round.

The sequences produced by the PRN generator described above possess the properties outlined in the description of a CSPRNG. The sensitivity to unique initial conditions is derived from the use and combination of multiple chaotic maps. The sequences' internal correlation and critical information content are then minimized by the use of the XOR and integer generation procedures. By selecting the byte used to generate the pseudo random integer near the end of the XOR product, minimal differences in initial conditions are expected to precipitate quickly into significant differences in the pseudo random sequence. The result is a quick and efficient CSPRNG.

As proof of principle, and as a reference for further applications, the PRN generator described above was implemented on an IBM compatible personal computer. Full documentation on the hardware and software specifications used are included in Section 4. All arithmetic is performed in 64 bit extended floating point precision.

Two chaotic maps were used, namely the *Bernoulli Shift*:

$$x_{n+1} = 2x_n \bmod 1$$

and the *Logistic Map*:

$$x_{n+1} = \lambda x_n(1-x_n)$$

whose properties are well studied. [5,6] Both possess simple recursive structures which make computer implementations quick and efficient. Iterates are XORed together, and the binary byte consisting of bits 48 through 55 of the 64 bit XOR product is extracted and converted into an integer. The initial information requirements consist of one 64 bit floating point seed for each map, a 64 bit λ value for the Logistic Map, and an 8 bit integer value l describing the number of iterations between subsequent values in the pseudo random sequence. This set of values is referred to as the *key*.

Using the CSPRNG presented above, a variation of the one time pad cryptosystem is achieved as follows: A message M of length L is separated into its component characters m_i. These characters are represented as integer values based on their position in the alphabet S_m. For illustration, the standard ASCII character set, which represents characters as 8 bit integer values, has been used. Using the method presented in the previous section a pseudo random integer sequence P of length L is generated, with eight bit integer components p_i. The ciphertext C is expressed in terms of its components c_i, defined as:

$$c_i = m_i \otimes p_i .$$

Decryption follows an entirely parallel scheme (See Fig. 1).

The Chaotic Dynamics XOR Cryptosystem (CDXC) enjoys a distinct advantage over the standard one time pad method. Each use of the traditional one time pad requires as much key material as plaintext, resulting in the key management problem discussed in the previous section. However, our variation of the one time pad requires a constant, relatively small amount of initial information for each message to be encrypted, regardless of its length. The initial information requirements of this particular application consist of three 64 bit values and one 8 bit value, or 200 bits of information. This translates to the secure distribution of 25 ASCII characters per encryption, effectively eliminating the key management problem associated with the traditional one time pad while retaining its security.

Encryption of a Plaintext Character

This chart represents the encryption of a plaintext character using the encryption scheme presented in section three. Due to the properties of the XOR operation, decryption is achieved by an identical mechanism. In decryption the input consists of a ciphertext element, c_n, while the output is a plaintext character m_n.

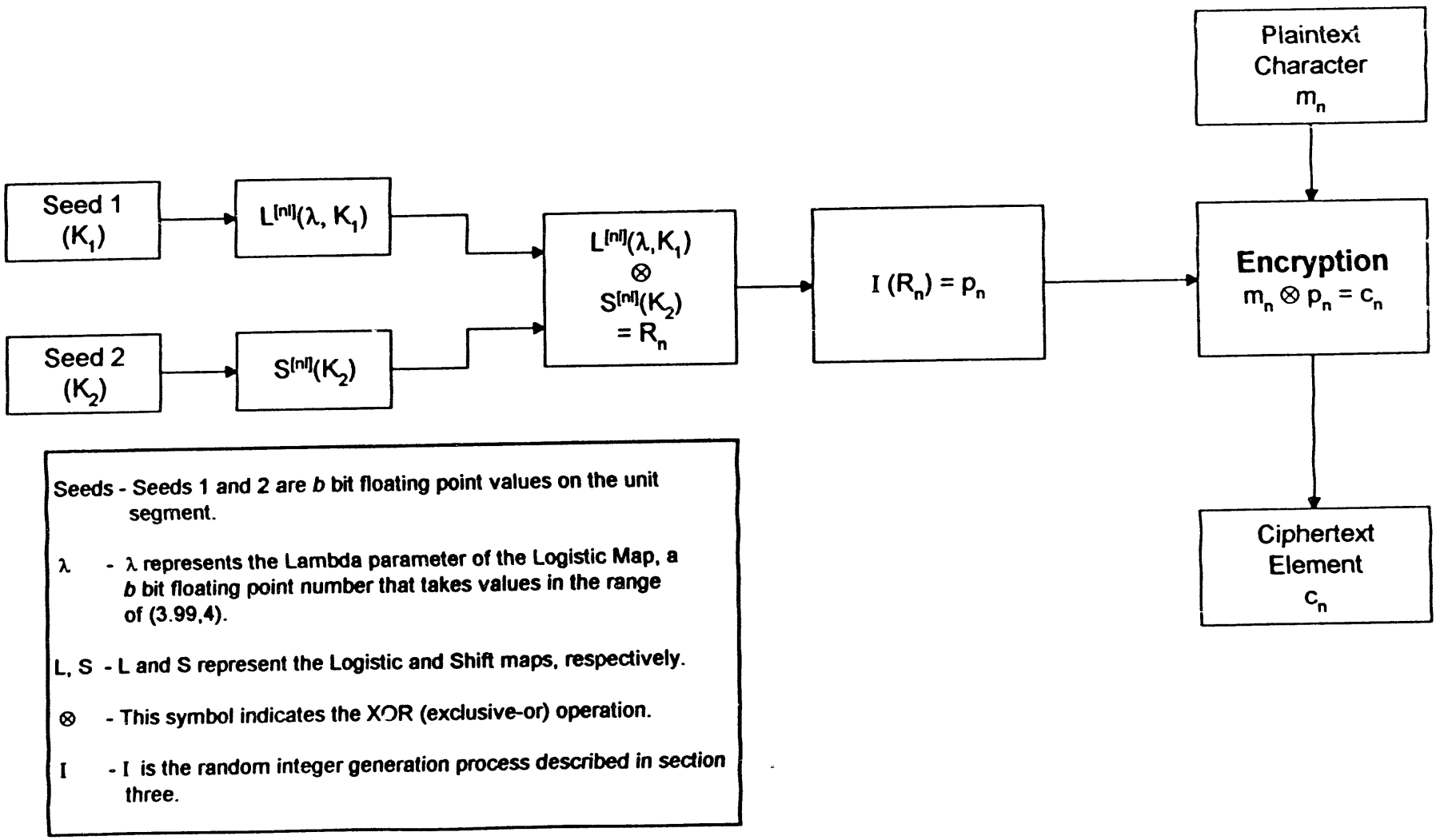

Figure 1

4. IMPLEMENTATION

To demonstrate the utility of the CDXC, a computer application was developed. The computer hardware used in the development of the DOS version of this program consists of a Northgate Computer Systems, Inc. 386 personal computer running Microsoft's MS-DOS 5.0. This machine makes use of the Intel 80386 - 25 MHz 32 bit processor and the Intel 80387 numeric coprocessor. The 80387 numeric coprocessor provides for the quick manipulation of floating point operations, and is capable of supporting an 80 bit extended precision floating point mode. Standard IEEE rounding modes are also supported, the default state of round to nearest was used in all floating point calculations. The source code was developed using Borland Turbo C++ version 3.0.

The core of CDXC is the encryption algorithm presented in the previous section. Additional routines were added to handle user input and file I/O. The program is a command line encryptor; when invoking it a source file is specified on the command line. This file serves as the input for encryption, the *plaintext*, and is overwritten by encrypted output, the *ciphertext*. As described before, a pseudorandom 8 bit value (one byte) is extracted from the XOR product of two chaotic iterates, and is XORed with a plaintext byte. The XOR value of the pseudo random and plaintext bytes is output as the ciphertext byte. Due to the properties of the XOR operation, encryption and decryption are identical functions. To decrypt a file, CDXC is invoked with the encrypted file specified as the source file. By supplying the proper initial conditions, the correct pseudo random sequence is generated and XORed with the ciphertext, reproducing the original plaintext.

The DOS version of this program has been extensively tested with various types of files, functioning equally well on both text and binary data. Furthermore, CDXC runs very quickly. Even on a low end platform like the 80386 computer used here, it outpaced several software implementations of the DES, with throughput on the order of 16k/sec (average). Note that this rate includes the delay caused by disk I/O, which was optimized through the use of multiple buffers but still requires a significant amount of time for large files.

A UNIX version of this program was developed, using vendor supplied compilers to compile the code on different workstations. An HP 9000/730 workstation was used as the primary development platform, with versions ported to IBM RISC/6000 580 and SGI Indigo (R3000) workstations. Ciphertexts were interchangeable between these three machines. A message encrypted on one machine was correctly decrypted on another, dispelling our initial fear that differences in architecture and consequently different methods of addressing floating point numbers on different workstations would prevent encryption and decryption across platforms. The high throughputs for

these workstations, displayed in the table below, makes this cryptosystem ideal for applications involving high speed transfer of data over networks and phone lines.

Platform / Processor	File Size (bytes)	DES Encryption Time (seconds)	CDXC Encryption Time (seconds)
386 - 25 MHz	519750	64.4	32.8
486 - 33 MHz	519750	23.0	7.96
IBM RISC/6000 580	519750	3.4	1.1
HP 9000/730	519750	4.4	1.9
SGI Indigo (R3000)	519750	*	6.7

* - No Version Available

5. DISCUSSION

The cryptosystem presented in Section 3 effectively removes the key management problem of the one time pad for a large number of practical applications, without decreasing its security. The benefits derived from employing chaotic maps in encryption are best evaluated when this cryptosystem is subjected to various methods of attack used by an enemy. Consider, for example, two trivial methods of attack: (i) Brute Force attacks and (ii) Key Guessing attacks. It is not difficult to demonstrate their inefficiency as long as we observe certain rules which must be applied to our scheme.

The term Brute Force Attack can describe any one of several types of attacks on a cryptosystem, most of which resort to an exhaustive search of some set of parameters intrinsic to that cryptosystem. A typical Brute Force Attack might attempt to decrypt a ciphertext by using every possible key, until the correct key is found. Such an attack is, in the case of CDXC, impractical due to the computational infeasibility of examining its large keyspace. However, while the chances are minimal, it is not inconceivable that such an attack might (very) rarely succeed. In order to maintain the security of the scheme *a unique key must be used for every encryption.* Doing so prevents the decryption of multiple ciphertexts in the extremely unlikely event of the enemy obtaining the key for a particular ciphertext.

A Key Guessing attack is employed when the enemy suspects a biased distribution of initial conditions over the keyspace. Such a situation might arise when the user of the cryptosystem chooses keys which fit a discernible pattern, such as common English words or obvious combinations of the date or time. In this situation, an enemy can make use of this knowledge to reduce the size of the keyspace to a practically accessible size. In order to avoid such attacks, *the key selection process must possess a uniform distribution over the keyspace.* If this rule is followed a Key Guessing attack becomes impractical.

While the two attacks presented above are by no means fully representative of the arsenal of an experienced cryptanalyst, they serve to define the two main guidelines which must be observed when employing this cryptosystem. If these rules are followed, we may defer to the proof of the one-time pad's security as an argument for the security of this scheme, but a final argument can not be made until the PRN generation method has undergone extensive analysis and attack.

Further examination of the properties of the cryptosystem and its associated PRN generator must begin with a careful scrutiny of the character of the chaotic maps. The quality of the PRN generator relies on whether or not these maps allow for the uniform distribution of bits over the eight bit address from which the pseudo random byte is removed. In addition to an examination of the quality of the maps used in the

PRN, the irreproducibility of its output must be verified. It is necessary to demonstrate, beyond simple arguments of plausibility, that reproduction of the output of the PRN generator is computationally infeasible in a reasonable amount of time, without the exact initial conditions used in the creation of that output. Also, periodic behavior in the PRN generator must be addressed. While periodicity has not yet been observed, its existence might prove to be a complication, possibly causing unwanted redundancies in the ciphertext. Similarly, the effect of finite precision arithmetic on the PRN generation process must be considered. Calculations involving finite precision are known to cause relatively small periodic loops in sequences of chaotic iterates. An examination of this phenomenon reveals that careful selection of map parameters may serve to minimize the effects of any precision-related periodicity.

The speed and security associated with this cryptosystem makes it ideal for use in any situation which demands these qualities. For example, the high throughput of our initial software implementation makes it an ideal candidate for use in the transmission of encrypted electronic mail across networks, encrypting "on the fly". A refined software application of this scheme, or a dedicated hardware implementation, would operate at speeds sufficient for the dynamic encryption of high speed data transfer, making the real time encryption of digital communications practical. The scheme's strength makes it equally useful for the encryption of high security documents and information. Such demands might arise due to the results of industrial espionage, electronic theft and laundering, or violations of network security. While this scheme is sufficient for applications requiring speed or security, its powerful combination of the two qualities makes it suitable for applications which demand both, offering a versatile alternative to iterated cryptosystems.

REFERENCES

1. D. Welsh, Code and Cryptography, Oxford University Press, Oxford, 1990.

2. D. E. R. Denning, Cryptography and Data Security, Addison-Wesley, Reading, MA, 1983.

3. National Bureau of Standards, Data Encryption Standard, U.S. Department of Commerce, FIPS pub. 46, January 1977.

4. E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, New York, 1993.

5. E. A. Jackson, Perspectives in Nonlinear Dynamics, Cambridge University Press, Cambridge, vol 1, 1989; vol 2, 1990.

6. S. Neil Rasband, Chaotic Dynamics of Nonlinear Systems, John Wiley and Sons, New York, 1990.

7. B. Jansson, Random Number Generators, PhD Thesis, Victor Petersons Bokindustri AB, Stockholm, 1966.

8. D. E. Knuth, The Art of Computer Programming, vol 2, Seminumerical Algorithms, Addison-Wesley, Reading MA, 1996.

9. M. Blum and S. Micali, How to Generate Cryptographically Strong Sequences of Pseudo Random Bits, SIAM J. Comp. 13, 850-863 (1984).

10. J.P.C. Kleijnen and B. Annink, Pseudorandom Number Generators for Supercomputers and Classical Computers: A Practical Introduction, European Journal of Operational Research 63, 76-85 (1992).

11. G. J. Chaitin, Randomness in Arithmetic, Scientific American, p. 81-85, July 1988.

12. M. Kendall and A. Stuart, The Advanced Theory of Statistics, Griffen, New York, vol 3, 1983.

INTERNAL DISTRIBUTION

1.	B. R. Appleton
2.	J. M. Barnes
3.	T. J. Burns
4.	J. D. Drischler
5.	C. M. Haaland
6.	D. T Ingersoll
7.	J. O. Johnson
8.	J. V. Pace
9.	R. T. Primm
10-14.	V. Protopopescu
15.	W. A. Rhoades
16.	R. C. Mann
17-21.	R. T. Santoro
22.	R. C. Ward
23-27.	V. Tolliver
28-32.	EPMD Reports Office
33-34.	Laboratory Records Department
35.	Laboratory Records ORNL-RC
36.	Document Reference Section
37.	Central Research Library
38.	ORNL Patent Section

EXTERNAL DISTRIBUTION

39. Prof. Roger W. Brockett, Harvard University, Pierce Hall, 29 Oxford Street, Cambridge, MA 02138

40. Prof. James N. Damon, Dept. of Mathematics, Room 331, Phillips Hall, University of North Carolina at Chapel Hill, Chapel Hill NC 27514

41. Prof. Donald Dudziak, Dept. of Nuclear Engineering, 110B Burlington Engineering Labs., North Carolina State University, Raleigh, NC 27695-7909

42. Commanding Officer, U. S. Army Missile Command, ATTN: AMSMI-RD-AC (Dr. Bruce Fowler), Redstone Arsenal, AL 35898-8174

43. Dr. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815

44-48. R. A. McNees, 321 W. Cameron, Chapel Hill, NC 27514

49. Prof. Sean Washburn, Dept. of Physics, Room 278, Phillips Hall, University of North Carolina at Chapel Hill, Chapel Hill, NC 27514

50. Prof. Ansel C. Mewborn, Dept. of Mathematics, Room 331, Phillips Hall, University of North Carolina at Chapel Hill, Chapel Hill, NC 27514

51. Marcia Smith, Columbia Cascade, Inc. 12020 Sunrise Valley Drive, Suite 200, Reston, VA 22091-3429

52. Thomas Reader, U. S. Army Vulnerability Assessment Laboratory, White Sands Missile Range, NM 88002-5513

53. Dr. Helena Wisniewski, VITA Inc., 1600 Wilson Blvd., Suite 500, Arlington, VA 22209

54. Prof. Neville Moray, Dept. of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801

55. Prof. Mary F. Wheeler, Dept. of Mathematics, Rice University, P. O. Box 1892 Houston, TX 77251

56. Office of the Assistanct Manager for Energy Research and Development, Department of Energy, Oak Ridge Operations, P.O. Box 62, Oak Ridge, TN 37830

57-58. Office of Scientific and Technical Information, P.O. Box 62 Oak Ridge, TN 37830