A Novel Compression and Encryption Scheme Using Variable Model Arithmetic Coding and Coupled Chaotic System

Ranjan Bose, *Member, IEEE,* and Saumitr Pathak

***Abstract*—Past research in the field of cryptography has not given much consideration to arithmetic coding as a feasible encryption technique, with studies proving compression-specific arithmetic coding to be largely unsuitable for encryption. Nevertheless, adaptive modeling, which offers a huge model, variable in structure, and as completely as possible a function of the entire text that has been transmitted since the time the model was initialized, is a suitable candidate for a possible encryption-compression combine. The focus of the work presented in this paper has been to incorporate recent results of chaos theory, proven to be cryptographically secure, into arithmetic coding, to devise a convenient method to make the structure of the model unpredictable and variable in nature, and yet to retain, as far as is possible, statistical harmony, so that compression is possible. A chaos-based adaptive arithmetic coding-encryption technique has been designed, developed and tested and its implementation has been discussed. For typical text files, the proposed encoder gives compression between 67.5% and 70.5%, the zeroth-order compression suffering by about 6% due to encryption, and is not susceptible to previously carried out attacks on arithmetic coding algorithms.**

***Index Terms*—Arithmetic coding, chaos, compression, encryption, symmetric key cryptography, variable model.**

I. Introduction

THE IDEA of combining data compression with encryption of data is a useful one since, in addition to an optimization of storage space required and transmission times needed, compression leads to a decrease in the redundancy in the plaintext, which makes the data more resistant to statistical methods of cryptanalysis [1]. Nevertheless, incorporating cryptographic features in a compression algorithm is a difficult exercise and often leads to a compromise between the amount of compression achieved and the amount of security incorporated [1], [2]. Also, computational resources today allow the cryptanalyst to carry out organized attacks with much success, and the time required to carry out brute force attacks, too, has been greatly reduced. From a strong cryptographic point of view, the security of data should be based on an algorithm where brute force attack has a minimum of 2^{128} choices in the search space [1]. Robust algorithms, promising sufficient data security, are, therefore, an ever-recurring need.

The existence of a possibly positive relationship between chaos and cryptography has been recently pointed out [3]–[7] and chaos, of late, has gathered attention as an attractive, almost fashionable, approach for devising new cryptographic algorithms. Characteristics of chaotic systems like ergodicity, mixing and sensitivity to initial conditions have been seen as analogous to and/or giving rise to confusion and diffusion, balance and avalanche property [8], [9], known properties of a good cipher [1]. Many cryptosystems based on chaos have been proposed [3], [10]–[17]. However, a number of chaotic cryptosystems have not been subjected to proper cryptanalysis and therefore cannot be considered to provide sufficient security; some have been shown to possess weaknesses or are vulnerable to cryptanalysis [18]–[26]. This has also given rise to widespread scepticism regarding chaos as having any relationship with cryptography. It has been noted in [7] that while initially the publications on chaotic encryption were discussed also in the community of applied cryptography, this interdisciplinary debate seems to have ceased.

Meanwhile, balanced and insightful suggestions have been proposed [8], [9], [17], [26], which recommend focus of future research on the relationships between chaos and cryptography, not the *ad hoc* design of new chaotic ciphers, proposed in haste and without proper testing and analysis. New structures of chaotic ciphers will be useful if novel ideas are provided with scope for future research. It should be noted that analog chaotic algorithms, based on synchronization, are not secure. Recent research has indicated that carefully considered and designed digital encryption algorithms based on chaos may be useful, being fast as well as cryptographically secure. Algorithms like the recently proposed pseudorandom bit generator (PRBG) based on a couple of chaotic systems (CCS) [9] and the chaotic video encryption scheme (CVES) [17] which combine the random behavior of conventional pseudorandom number generators (PRNGs) and chaos-based RNGs, may thwart cryptanalysis on both the fronts: chaos theory-based methods as well as conventional cryptanalysis. Popular RNGs like the linear congruential generator, for which fast software implementations exist, suffer from a short cycle length and are not considered robust from a strong cryptographic point of view [27], [28]. CVES proposed in [17] has been found to be insecure against chosen-plaintext attack by Li and enhanced in [29]. Currently available excellent RNGs consist of various implementations and variations of the multiple recursive generators [30], [31],

Manuscript received May 23, 2004; revised December 22, 2004 and May 3, 2005. This work was supported by the SERC, Department of Science and Technology, New Delhi. This paper was recommended by Associate Editor S.-G. Chen.

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Delhi, New Delhi 110 016 India (e-mail: rbose@ee.iitd.ac.in).

Digital Object Identifier 10.1109/TCSI.2005.859617

and the linear feedback shift registers (LFSRs), among others, and the CCS PRBG compares well with them.

Arithmetic coding [32]–[35], in its essence, assigns to each possible character a range, the width of which reflects the frequency of occurrence of the symbol. This conceptual space, consisting of all such ranges arranged cumulatively, is shrunk consecutively to uniquely represent the input symbols. Each consecutive symbol causes the current range boundaries to shrink to accurately represent its occurrence, using the range allotted to it, and after each revaluation of the current limits of the range, the entire symbol space, having been updated to account for the new symbol, is mapped into this new range. This is accompanied by an increase in the precision bits of the variables used to represent the boundaries. In practice, a window is used to represent the current range boundaries, to prevent computational overflow, while the most significant bits are continuously shifted out and filed. Accurate prediction of symbol probabilities is essential to the efficiency of compression.

Past research in the field of cryptography has not given much consideration to arithmetic coding as a feasible encryption technique and, while arithmetic coding has still been the focus of a lot of intense research, studies by [36] and [37] have proven compression-specific arithmetic coding, both adaptive and fixed-model, to be largely unsuitable for encryption. This is in spite of the large size of the model that arithmetic coding works with. Adaptive modeling, in particular, offers a huge model, variable in structure, and as completely as possible a function of the entire text that has been transmitted since the time the model was initialized. This serves as a huge key which can make decryption arbitrarily difficult [38]. But the scheme has been shown to be vulnerable to organized cryptanalysis [37]. Nevertheless, the difficulty in resynchronization arithmetic codes [39], [40] offers motivation for using arithmetic codes for encryption. In particular, joint compression and encryption, sharing the computation cost, has been proposed in [40].

The focus of the work presented in this paper has been to incorporate the recent results of chaos theory, proven to be cryptographically secure, into arithmetic coding, to devise a convenient method to make the structure of the arithmetic coding model unpredictable and variable in nature, and yet to retain, as far as is possible, statistical harmony, so that compression is possible. This paper is organized as follows. The system model has been introduced in Section II along with a brief discussion of the issues related to the design of good chaotic ciphers and arithmetic coding. The proposed idea has been put forward and briefly discussed in Section III. Simulation results on the strength of compression and encryption have been described in Section IV. The strength of the encryption has been further discussed in Section V. Conclusions are given in Section VI.

II. System Model

The methodology being proposed in this paper seeks to effectively combine the unpredictable behavior of the logistic function with a variable-model adaptive arithmetic coder. We consider a model for carrying out zeroth-order adaptive arithmetic coding.

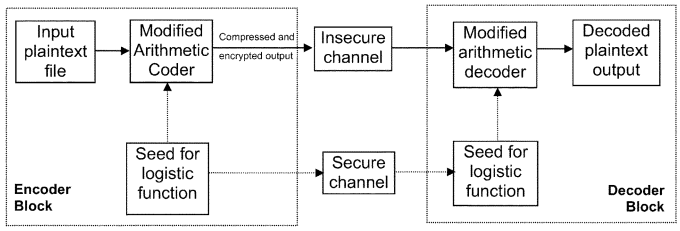


Fig. 1. System model.

The encoder block (see Fig. 1) consists of the modified arithmetic coder, which uses the secret information (the key which consists of the initial condition/parameters for the chaotic map) to process the input plaintext file, giving compressed and encrypted output. The information from the encoder block uses the insecure public channel to transmit the coded data to the decoder block, while a private, secure channel is used to transfer the key for the chaotic map. The modified arithmetic decoder recreates the encoder model using the encoded data and the key and carries out decoding in a similar fashion.

While designing good ciphers based on chaotic systems there are certain problems that should be considered. Chaotic maps, when realized in finite precision, have vastly different discrete dynamics than the corresponding continuous ones [18], [19], [25], [41]. This dynamic degradation, a departure from the theoretical behavior of chaotic systems, gives rise to problems like short cycle-length, nonideal distribution and correlation, etc. during their digital realization, which reduce cryptographic strength if not considered during the design of the cryptosystem. Several remedies have been proposed as solutions to this problem. They include the use of higher finite precision [18], perturbation-based algorithm by pseudorandom number [42], [43] and the use of multiple chaotic systems [9]. Recently, a technique for PRBG has been proposed [9] based on CCS . When the coupling chaotic systems satisfy certain requirements, the bitstream generated using this method has excellent cryptographic properties, including long cycle length and good statistical characteristics. Methods have been suggested to extend the cycle length even further [43]. The logistic map and the piecewise-linear chaotic maps (PLCM) can be used as the coupling systems for the CCS PRBG.

Complicated chaotic systems entail the use of floating point arithmetic, which affects speed of the algorithm as well as hardware and software realizability. Fixed-point arithmetic and simple chaotic systems are recommended for speeding up encryption and to ensure simple realization by hardware and software [9], [17], [25]. Because chaotic systems are deterministic systems, there are some tools in chaos theory to discern chaos. Once an intruder finds some information about the chaotic systems from their orbits, he might use such information to lessen the complexity of finding the secure key. With the use of multiple chaotic systems, the cryptanalysis of chaotic ciphers will be more difficult since the output is determined by many different mixed chaotic orbits. Another characteristic of a good chaos-based cipher is the chaotic-system-free property. This implies that the encryption algorithm is not formulated around a particular chaotic map but can work with a number of chaotic systems.

In our algorithm, we make use of the CCS PRBG as a kernel for the generation of a random bitstream. This bitstream is used to bring modifications in the model of the arithmetic coder. The CCS PRBG addresses all the problems noted above. It possesses excellent statistical and cryptographic properties: 1) balance on $\{0, 1\}$; 2) long cycle-length; 3) high linear complexity approximating to half of the cycle-length; 4) δ-like auto-correlation; v) cross-correlation near to zero; vi) chaotic-system-free. In addition, it can be used in fast stream ciphers. Stream ciphers designed around CCS PRBG using the PLCM and fixed-point arithmetic reach speeds of 9 Mbps [9], which is an acceptable speed in many secure applications. This speed also compares well with other modern RNGs [30], [31].

Consider two different chaotic maps $M_1(x_1, p_1)$ and $M_2(x_2, p_2)$. For these maps, iterations may be represented as

$$\begin{aligned} x_1(i+1) &= M_1(x_1(i), p_1) \text{ and } x_2(i+1) \\ &= M_2(x_2(i), p_2). \end{aligned} \tag{1}$$

Here, $x_1(0)$ and $x_2(0)$ are the initial conditions and p_1 and p_2 are the control parameters. The jth pseudorandom bit, b_j, is defined as

$$b_j = \begin{cases} 1, & x_1(i) > x_2(i) \\ \text{No output}, & x_1(i) = x_2(i) \\ 0, & x_1(i) < x_2(i). \end{cases} \tag{2}$$

If no output takes place, the systems (1) are iterated one step until an unequal output is obtained. This will not result in any loss of synchronization because the coupled chaotic system based CCS PRBG module can be run at a much higher rate than the arithmetic coding module. Successive bits are combined to form $(\log n)$-bit and $(\log t)$-bit words, as explained below (the logarithm is to the base 2). Here n denotes the number of symbols in the alphabet and t is the threshold parameter used for controlling the deviation in symbol statistics. An example of a PLCM, which was analyzed in [41], is

$$M(x, p) = \begin{cases} (x/p, & X \in [0, p) \\ (x-p)/(1/2-p), & X \in [p, 1/2] \\ M(1-x, p), & X \in [1/2, 1]. \end{cases} \tag{3}$$

The logistic map can be represented as:

$$M(x, p) = px(1-x). \tag{4}$$

This equation takes a period doubling route to chaos as the system parameter increases from 1 to 4. Chaotic behavior appears for values of $p > 3.57$, the "point of accumulation," although there are points of local stability/periodicity in this region too [12], [44], [45].

To further enchance cryptographic properties of the generated bitstream, a perturbation-based algorithm is suggested in [42]. Two PRNGs are used to generate uniformly distributed signals, which are used to perturb, say lb, lowest bits of $x_1(i)$ and $x_2(i)$, with intervals Δ_1 and Δ_2. The maximal length linear feedback shift registers (m-LFSR) and the linear congruential generators are suggested PRNGs for this purpose. Here $lb = T$, to retain the characteristics of the chaotic systems, where T is the finite computing precision. Because of their sensitivity to initial conditions, the chaos signals are driven in a very complex way, in spite of the small size of the perturbing signal. The combination of digital chaos and the pseudorandomness of PRNGs is seen to make both chaos-theory-based and conventional cryptanalysis difficult.

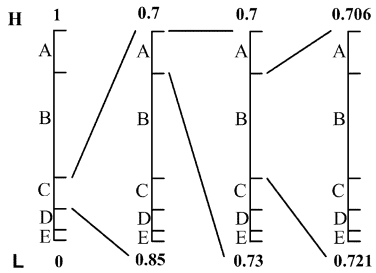


Fig. 2. Encoding of "CAB" in a 5-symbol alphabet.

If two m-LFSR-s are used as the perturbing PRNG-s, whose degrees are L_1, L_2, and perturbing intervals are Δ_1, Δ_2. Then the cycle-length of $x_1(i)$ and $x_2(i)$ are

$$\sigma_1 \Delta_1 (2^{L_1} - 1) \text{ and } \sigma_2 \Delta_2 (2^{L_2 - 1}) \tag{5}$$

where σ_1, σ_2 are two positive integers [42]. So the cycle-length of b_i will be

$$\text{lcm}(\sigma_1 \Delta_1 (2^{L1} - 1), \sigma_2 \Delta_2 (2^{L2} - 1)). \tag{6}$$

Parameters can be chosen so that the cyclical length will be

$$\text{lcm}(\sigma_1, \sigma_2) \cdot \Delta_1 \cdot \Delta_2 2^{L1+L2} \tag{7}$$

as has been demonstrated in [42].

Arithmetic coding is a compression technique which assumes an explicitly probabilistic model of the source [46] and carries out very nearly optimal compression for the given probability estimates [30]. Separation of the coder from the modeling process in arithmetic coding provides a high degree of flexibility to the technique, allowing us to modify the modeling process for encryption purposes, without changing the basic encoding algorithm. The arithmetic coding algorithm is explained below with the help of a small example.

Consider an alphabet {A, B, C, D, E} with known probabilities {0.2, 0.5, 0.15, 0.1, 0.05}. The conceptual space [0, 1) is divided among A, B, C, D and E consecutively with A occupying [0, 0.2), B occupying [0.2, 0.7), C occupying [0.7, 0.85) and so on. The current interval $[L, H)$ is initialized to the entire conceptual space, but is shrunk to the current symbol's range as a symbol is encoded. A C occurring as the first character shrinks $[L, H)$ to [0.7, 0.85). After each occurring character, the entire range divisions are mapped to the current interval so that when the next character is encountered, it results in further shrinking of the interval. Therefore, an A, followed by a B shrink the current interval $[L, H)$ to [0.7, 0.73) and then to [0.706, 0.721) (refer to Fig. 2).

Since a smaller interval requires a larger precision to be represented, symbols with larger range assigned to them cause less shrinking and comparatively lesser increase in the precision bit length of the variables L and H, which enables compression to take place. It is evident that, in a real life input, before long the current interval would require especially long floating point

numbers to be represented accurately. Therefore, a few additional steps are carried out to avoid overflow of the variables L and H. Integer arithmetic, bit-shifting and interval doubling are generally employed in implementations to avoid the problem of overflow without affecting compressive performance.

While it can be seen that arithmetic coding requires multiplications and additions at every step, modified and optimized implementations achieve high encoding rates. The basic implementation of zeroth-order arithmetic coding given in [32] encoded at a rate of 4.83 MB/min (644 kb/s). Improved coders, with shift/add operations replacing some multiplications, encode at a rate of up to 10.59 MB/min (1.41 Mb/s). On architectures offering integer multiply and divide operations in hardware, much higher speeds of up to 25.5 MB/min are achieved [35].

While the example above had estimates of the symbol probabilities before the encoding process, for single-pass operation an adaptive scheme is implemented, where symbol probabilities are updated to account for each new symbol encoded. Also, while the zeroth-order arithmetic coder compresses the input optimally close to the entropy of the input, much higher compression can be achieved by using higher order, highest context modeling.

III. Variable Model Arithmetic Coding

The basic idea is to shuffle the probability of input symbols chaotically, i.e., dynamically change the probabilistic model, prior to performing the arithmetic coding. The shuffling (reordering) is done by the slot filling algorithm described in the first part of this section. Clearly, when we interchange the ordering of the symbols, it results in assuming a pseudostatistics of the incoming symbol stream. The latter half of this section assesses the deviation in the statistics in terms of a threshold parameter. This threshold parameter can be used to tradeoff the security with respect to compression.

We carry out two basic operations to make the arithmetic coding secure and the decoding completely key-dependent. Using the CCS PRBG, the key consists of $x_1(0)$ and $x_2(0)$, the initial conditions for the two chaotic maps, and p_1 and p_2, the control parameters. In the case of the logistic map, where the control parameters should be 4.0 for true chaotic behavior. For higher security requirements, other methods as suggested in [9] can be used. In this paper we have used the Logistic map based CCS PRBG because of the ease and speed of implementation. The Logistic map is also easy to translate into hardware. The objective of this paper is to demonstrate that coupled chaotic systems can be used effectively for joint encryption and compression. It should be noted that logistic map's probability density function is not uniformly distributed. It remains to be investigated whether the results change significantly by using a different map, such as a PWLCM [9].

Before the actual coding and transmission of encoded-encrypted bits, we break up the conceptual symbol space (which is the statistical model to be used by arithmetic coding), between the different symbols, using an algorithm based on the chaotic bitstream. The generation of chaotic S-Box has also been proposed in [14], [15]. For an n symbol alphabet, the chaotic bitstream generator is iterated $\log n$ times, to obtain a $(\log n)$-bit word, for each symbol until the output lands in an empty slot. The slot is then assigned to the current symbol and the process repeated for each remaining symbol in the *ascii* table.

Considering a random function, uniform over its entire range, the first slot may be filled in the first attempt itself, with expectation of number of attempts 1 and no more. All other slots may be filled in any number of attempts, the probability of which can be summed as an infinite progression.

For an n symbol alphabet, the expectation value of the number of attempts is given by $E = \sum$ (no. of attempts) (probability of no. of attempts). For the first slot, E_1 is 1. E_2 can be written as

$$E_2 = 1\left(\frac{n-1}{n}\right) + 2\left(\frac{n-1}{n}\right)\left(\frac{1}{n}\right) + 3\left(\frac{n-1}{n}\right)\left(\frac{1}{n}\right)^2 + \cdots. \quad (8)$$

In general, for the ith case, the expression for E_i looks like

$$\begin{aligned} E_i &= 1\left(\frac{n-i+1}{n}\right) + 2\left(\frac{n-i+1}{n}\right)\left(\frac{i-1}{n}\right) \\ &\quad + 3\left(\frac{n-i+1}{n}\right)\left(\frac{i-1}{n}\right)^2 + \cdots \\ &= \left(\frac{n}{n-i+1}\right). \end{aligned} \quad (9)$$

The expected value of the total number of attempts for all n slots is given by

$$\sum_{i=1}^{n}\left(\frac{n}{n-i+1}\right). \quad (10)$$

For a 256 symbol alphabet, the number of iterations come out to around 1567. In the case of the logistic equation, if used without a proper randomiser to make its distribution uniform, the number of iterations range from 1800 to 2200 as the seed and the coefficient are varied. Therefore, the algorithm is implementable in real time, as is. Also, the logistic map is a surjective function and has perfect chaotic properties only for p near 4.0, and we found that for values of coefficient p below 3.997, initially 1, and then progressively higher number, of the slots remain unfilled and the algorithm loops indefinitely. This behavior is explained by the attractor diagram for the logistic map. Although p equal to 3.57 is the point of onset of the chaotic realm, there is a rich interleaving of points of local stability and chaos for p lying between 3.57 and 4 [45].

Another method of filling up the slots is to generate a pseudorandom number α in [0, 1) and then assign the ith symbol to the $(\lceil \alpha n_i \rceil)$th unoccupied slot, where n_i is the number of all unoccupied slots. With such an algorithm, the shuffling can be finished after n chaotic iterations.

Next, we consider an alphabet of n different symbols, $a_1, a_2, a_3, \ldots a_n$. This sequence of symbols is not arranged in *ascii* order, rather the ordering is obtained by the algorithm given above. Let the number of different symbols that have occurred in the input is l. When the algorithm has run for a long duration, $l = n$. The threshold parameter being used in

the algorithm is $t(\leq n)$ and denotes the number of symbol frequencies that will be used for reshuffling. Consider a symbol from an i.i.d. source, $s_k \in \{a_1, a_2, \ldots a_n\}$ that is to be encoded. If we take $c_1, c_2, c_3, \ldots c_n$ as the frequencies of occurrence for the symbols $a_1, a_2, a_3, \ldots a_n$, then the probability estimate for the kth symbol is $(c_k/\sum_{i=1}^{n} \tilde{c}_i)$, where c_k is the *a priori* probability of the kth symbol, s_k, of the sequence, while $\tilde{c}_i$ is the *a posteriori* probability of the ith symbol of the alphabet. This information is sufficient to carry out arithmetic encoding for the symbol s_k.

Next, we consider the sequence of sorted frequencies $c_1^o, c_2^o, c_3^o, \ldots c_n^o$, with $c_i^o \geq c_{i+1}^o$ for $1 \leq i \leq n$, with each element corresponding to the frequencies $c_1, c_2, c_3, \ldots c_n$. $a_1^o, a_2^o, a_3^o, \ldots a_n^o$ is the sequence corresponding to $a_1, a_2, a_3, \ldots a_n$, with each element of the sequence a_i^o having frequency count c_i^o for $1 \leq i \leq n$. If c_j corresponds to $c_{j'}^o$ in the sorted sequence then we define

$$\mathbf{N}' = \{c_{j'-t}^o, c_{j'-t+1}^o, \ldots, c_{j'-1}^o, c_{j'+1}^o, \ldots, c_{j'+t-1}^o, c_{j'+t}^o\} \tag{11}$$

and we denote as N the t_{eff} elements of the set $\mathbf{N}'$ which are closest in index distance to $c_{j'}^o$. Here,

$$t_{\text{eff}} = \min(l-1, t) \tag{12}$$

where t, referred to as threshold, is a parameter which controls the amount of deviation in statistics. Elements of the set $\mathbf{N}'$, as seen, belong to the frequency-sequence $c_1^o, c_2^o, c_3^o, \ldots c_l^o$, while the frequencies-sequence $c_{l+1}^o, c_{l+2}^o, \ldots c_n^o$, corresponds to symbols yet to occur.

The shuffling of the frequencies is done as follows.

1) Partition the interval [0, 1], into t_{eff} subintervals and select the x_kth partition. Let the corresponding element of the set $\mathbf{N}'$ be $c_{x_k}^o$, where $c_{x_k}^o$ is the sorted frequency corresponding to the symbol be $a_{x_k}^o$.
2) Interchange the two symbols $a_{x_j}^o$ and $a_{x_k}^o$, in the sorted sequence $a_{x1}^o, a_{x2}^o, a_{x3}^o, \ldots a_{x_n}^o$, which causes a reversal in the frequency counts of the two symbols, with $c_{x_j}^o$ now being the frequency count for symbol $a_{x_k}^o$ and $c_{x_k}^o$ being the count for $a_{x_j}^o$.
3) Change the relative location of both symbols in the conceptual space so the output corresponding to the next recurrence of one of these symbols changes. Since the statistical model for the arithmetic coding has now changed, for the next occurrence of the symbols $a_{x_k}^o$ and $a_{x_j}^o$, the probability estimate is more than the true value for one and less for the other, or the reverse, depending on whether $c_{x_k}^o \leq c_{x_j}^o$, or $c_{x_k}^o \geq c_{x_j}^o$, which, in its turn, depends on x_k.
4) Get the next symbol to be encoded and go to step 1).

It should be noted here that the initial symbol distribution, which is not based on the ascii character sequence, is itself chosen arbitrarily. The schematics of the encoder has been given in Fig. 3.

While for the zeroth-order modeling of the symbols the procedure above is followed, for higher order, higher context modeling, $c_{j'}^o$ and $c_{x_k}^o$ are also exchanged, an operation which takes

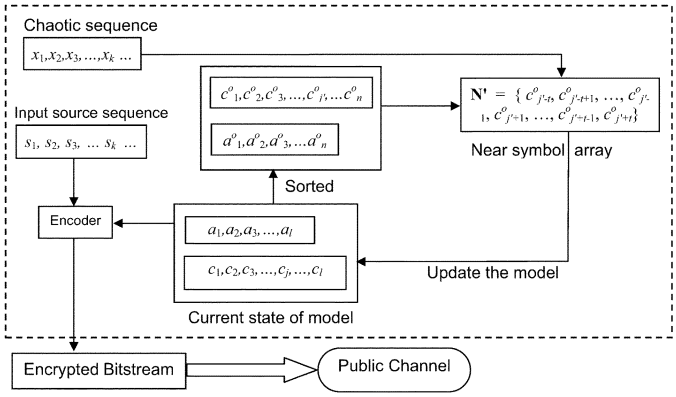


Fig. 3. Schematic of the encoder.

on an average $\log(n-1)$ time using the Fenwick tree structure [47], if the difference in $c_{j'}^o$ and $c_{x_k}^o$ is greater than a predetermined value. This should be done to prevent greater statistical deviation in the model due to the skewed statistics of the higher order model.

The algorithm can be used for transmission of individual messages, each with its own key. It can also be employed, for some purposes, as a continuous transmission engine. Here, use can be made of the synchronization between the transmitter and receiver which is necessary for correct decoding of the information. A symbol, say a nonoccuring one, may be fixed as a synchronization indicator, and may be encoded and sent after pre-determined intervals. This may help in data authentication and in detecting errors during transmission. Since this symbol can be made part of the key, and thus removed from public knowledge, it may not help in cryptanalysis.

IV. Simulation Results on Compression and Strength

For testing the proposed technique, we have used files from the Calgary Corpus (ftp.cpcs.ucalgary.ca/pub/projects/text.compression.corpus). The design parameters are $p = 4$ for both the chaos functions (logistic maps) in the CCS PRBG and threshold parameter lies in the range of $6 \leq t \leq 10$. A 10-bit accuracy of the digital CCS PRBG has been used for the simulations. The file paper5 is a normal English text file of size 12 kB with 92 distinct characters and a zeroth-order character entropy of 4.9818, which means the optimal zeroth-order compression for the file would be about 63%. The file obj1 of size 21 kB consists of executable code and is comparatively more difficult to compress because of a more uniform distribution of symbols. It has 256 distinct characters, a character entropy of 5.9482, and can be zeroth-order compressed optimally to about 75%. The shuffling of the frequencies has been carried out as described in the previous section. Frequencies for the alphabet have been stored using the fenwick tree structure, which allows operations in $\log n$ time [47]. In order to see the correlation between compression and frequency deviation, we define

$$\Delta f_o = \sum (|f_i - f_i'|)^2 \tag{13}$$

where f_i is the actual frequency and f_i' is the deviated frequency for the ith symbol. In general, Δf may represent the variance

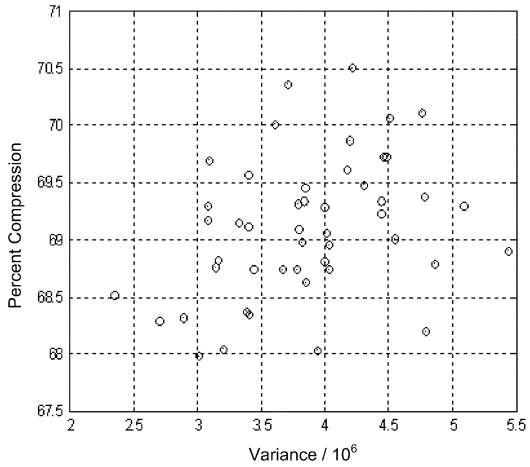


Fig. 4. Compressive performance of the zeroth-order arithmetic coder versus variance of symbol probabilities of the deviated distribution from the original distribution for the file paper5 (12 kB). The plot has been obtained by varying the seed values for the logistic map.

between two frequency distributions. We have used the logistic map for carrying out simulations.

The nature of the encoding algorithm may indicate an inverse relation between the degree of compression achieved and Δf_o. Fig. 4 shows a plot of percent compression versus variance for a typical text file (paper5). The initial condition for the chaotic map has varied to plot the graph. No direct correlation between compression achieved and the variance in the original and deviant symbol probability distribution can be seen. This is attributed to the fact that even though variance between the two variables has increased, it cannot be accurately predicted whether a symbol is being represented by extra probability space or is being under-represented, i.e., whether the symbol is being coded using lesser number of bits or greater number of bits than the optimal. Therefore, a combination of over-compression and under-compression occurs, causing greater than usual compression for certain symbols, while compression of other symbols suffers. Overall, there is not left much direct relation between Δf_o and compression achieved, while the strength of the encryption cannot be completely accounted for or measured through the above variables.

The degree of compression undergone by a text file has a relation with both the contents of the file and the initial condition for the chaotic map. Compression on a file varies as the seed value is varied, both in precision and in magnitude. Fig. 5(a) and (b) depict plots between the percentage compression achieved and the initial seed for the logistic function for two files. Seed values equally placed on either side of 0.5 yield similar distribution due to symmetry of the logistic map, while a seed value of 0.5 itself yields a constant series. Seeds after 0.5, including 0.5, have therefore been omitted. This is an interesting result. The performance is actually dependent on the initial condition of the logistic map. The spread is about 3% in the compression achieved. It is found that for the file paper5, maximum zeroth-order compression of about 67.65% was achieved when the seed value was 0.33, while the file obj1 was zeroth-order compressed at most to 80.4% when the seed value was 0.07. Thus, it is seen that in the modified zeroth-order arithmetic coder compression suffers, on

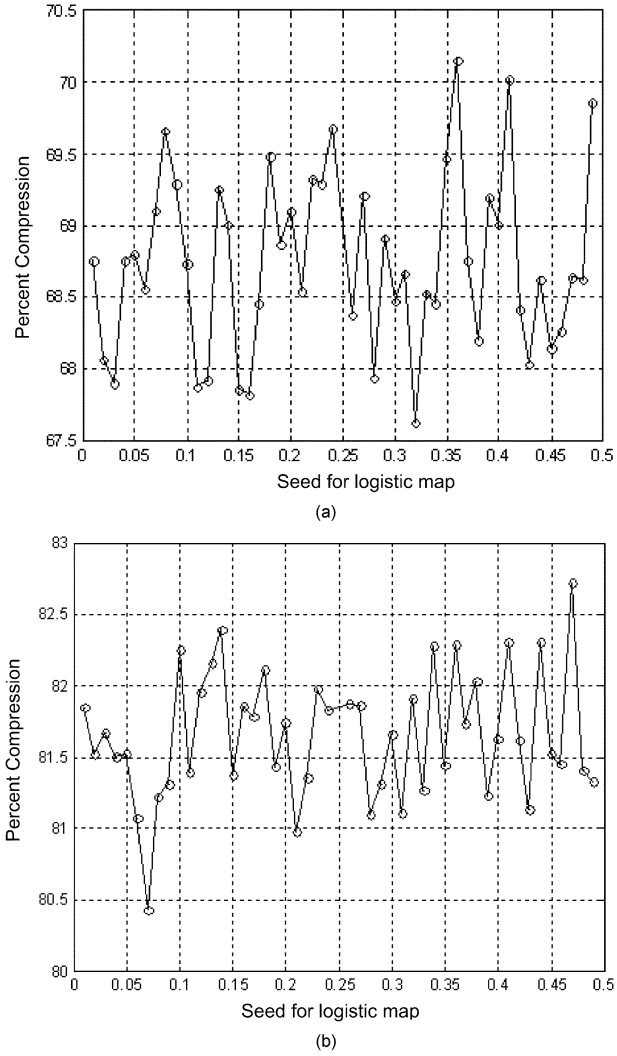
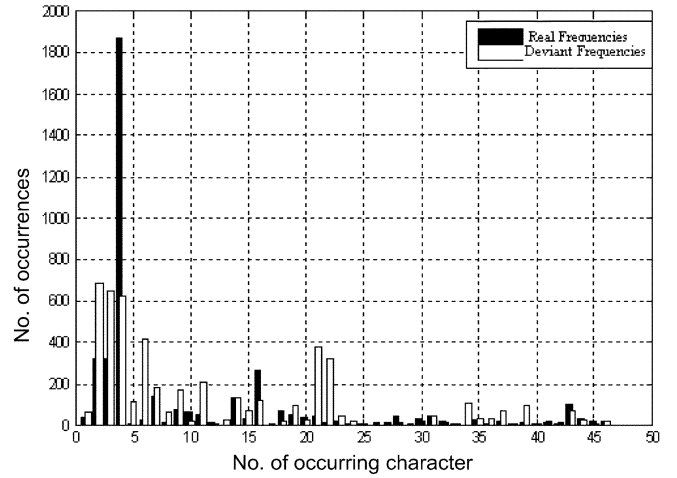


Fig. 5. Percentage compression achieved by the modified zeroth-order arithmetic coder as the seed for the logistic map is varied between but not including 0 and 0.5 for file (a) paper5 (12 kB) and (b) obj1 (21 kB).

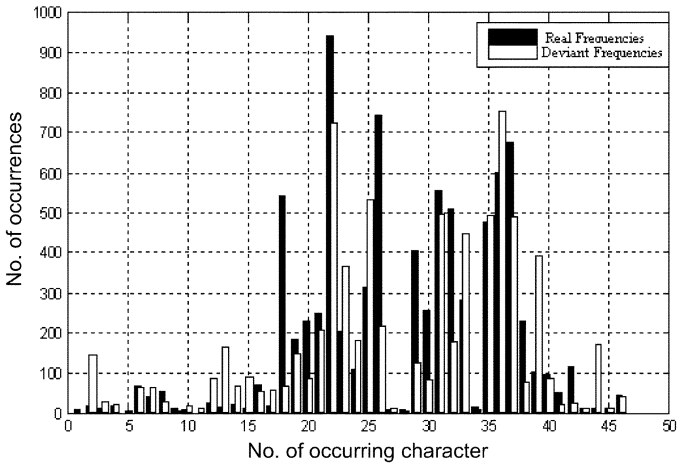
an average, by about 5.75% in the case of file paper5 and by about 6.5% in the case of file obj1. The threshold parameters for paper5 and obj1 were $t = 7$ and $t = 8$ respectively. It is seen that compression suffers by about 6% in the modified zeroth-order arithmetic coder algorithm. By increasing the order of the model, and storing symbol statistics in higher contexts, much higher compression is expected.

Next, we check the actual deviations in frequencies undergone by the symbols as a result of the swapping of symbol space. In Fig. 6(a) and (b), we have depicted the deviations in symbol frequencies at the end of the coding stage. The results are for the file paper5. The first 46 of the 92 different characters that occur in the file have been shown in Fig. 6(a) while the second half of the characters have been shown in Fig. 6(b). It can be noticed that some symbols have been under-represented while others have been over-represented. This is unfavorable for the cryptanalyst since statistical information of the symbols does not help cryptanalysis anymore.

To track the model as more characters are encoded, we follow the symbol swappings undergone by the $\langle \text{space} \rangle$ character. In Fig. 7(a), we have depicted the symbol slots among which

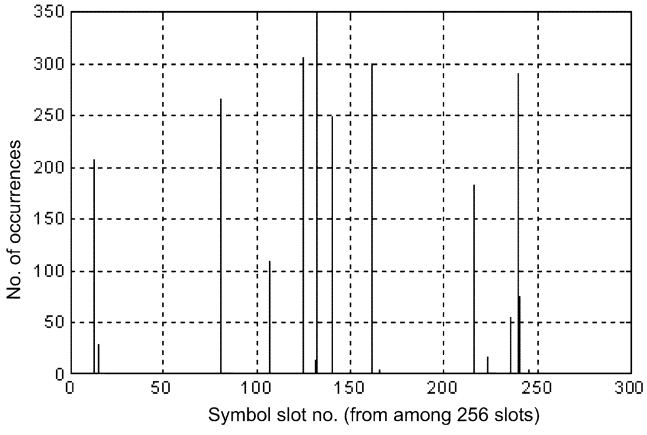


(a)

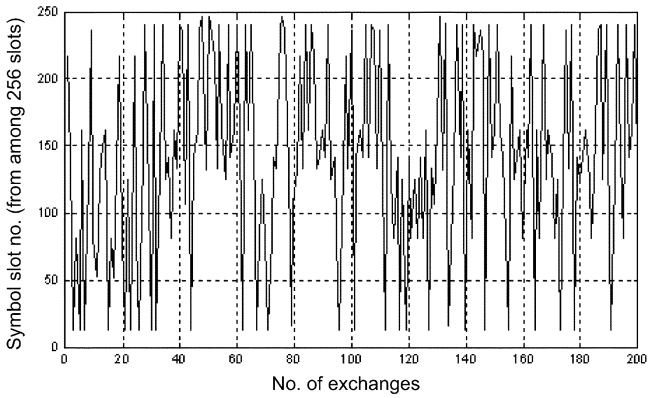


(b)

Fig. 6. Symbol frequency deviations in (a) first half of occurring symbols (b) second half. The results are for file paper5, which has 92 occurring characters. The filled bars represent the real frequencies while the empty bars represent the deviant frequencies.



(a)



(b)

Fig. 7. (a) Frequency of appearance of chr $\langle \mathrm{space} \rangle$ in various symbol slots as a result of exchanges among symbols. (b) The sequence of exchanges of the chr $\langle \mathrm{space} \rangle$ as it lands in the different symbol slots. The first 200 exchanges are shown.

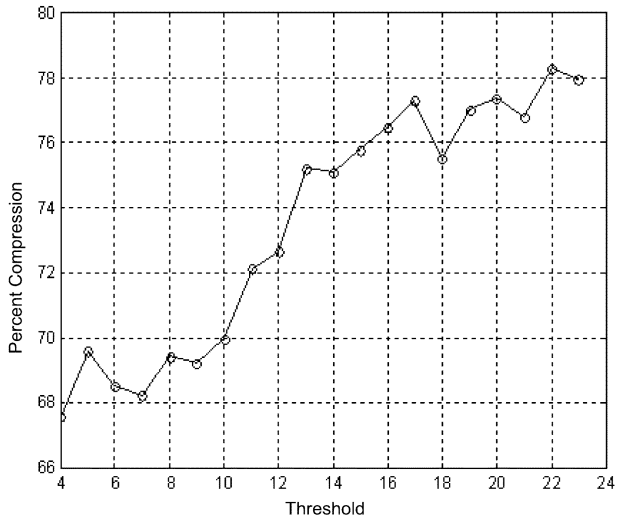


Fig. 8. Compressive performance of the zeroth-order arithmetic coder as the threshold parameter in the algorithm is increased. The threshold parameter controls the amount of deviation and unpredictability in symbol probabilities.

symbol 32, the $\langle \mathrm{space} \rangle$ character, is exchanged. Such slots are not more numerous than the ones shown since symbols are restricted to being exchanged only among their (extended) frequency-neighbors. In Fig. 7(b), we have shown how the $\langle \mathrm{space} \rangle$ character gets exchanged between the different slots. The first 200 exchanges have been shown. Both results are for the file paper5. The unpredictability which is evident from the graphs makes the variable nature of the model extremely difficult to track. This ensures that even known or chosen plain-text attacks can ascertain no useful information about the model, since no aspect of the statistical model is unchanging. In another test, with increase in the threshold parameter, t, the compression was found to suffer, as shown in Fig. 8. This parameter may be set to an optimum value for greatest compression. The proposed algorithm has been tested for reasonably large file sizes (up to 1 MB). It was found that beyond a certain file-size (about 1 kB) the performance (degree of compression) saturates.

V. Discussion on Strength of Encryption

When considering the strength of the encryption, it should be noted that, while the symbol sequence $a_1^o, a_2^o, a_3^o, \ldots, a_n^o$, corresponding to the sorted frequency-sequence $c_1^o, c_2^o, c_3^o, \ldots, c_n^o$, is used to carry out symbol exchange, which causes the symbols to depart from their real frequencies, the arithmetic code for a symbol is generated with the symbol space breakup as in the

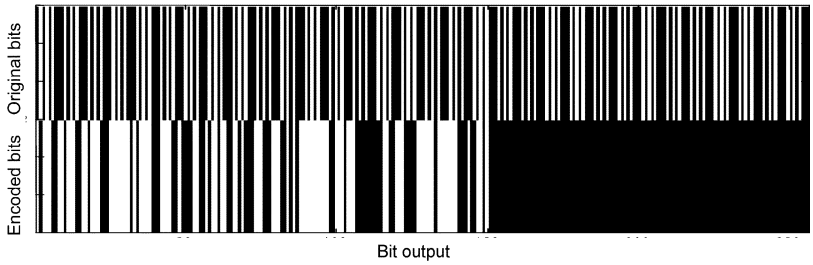


Fig. 9. Original and encoded (compressed) bit pattern for repetitive instances of character "T" (32 instances).

sequence $a_1, a_2, a_3, \ldots a_n$. On the other hand, further statistical prediction for symbol exchanges is based on the sorted frequency sequence $c_1^o, c_2^o, c_3^o, \ldots c_n^o$. This exercise encourages more chaotic exchanges between symbols and their frequency counts in the sequence $a_1, a_2, a_3, \ldots a_n$. The most vulnerable moment, cryptographically, for the model is when it is initially started. A low value of l, combined with known plaintext, may help determine the relative ordering of some symbols, by comparing closely matched outputs of strings like "bbbbb" and "bbbba," as described in [36]. As more characters are inserted into the encrypting engine, the symbol spaces get swapped, more and more chaotically, and no useful output which can help determine ordering can be obtained. Since the current symbol, s_k, that is inserted into the engine is immediately swapped with one of the symbols in $\mathbf{N}'$, using the value x_k to determine the character for exchange, known and chosen plaintext attacks will not work with the system. The proposed algorithm is resistant to chosen plaintext attacks proposed in [36] because the model *dynamically* reorders the frequency of the input symbols according to the coupled chaotic system. The ordering will be different each time the same (known or chosen) plaintext is fed into the encryption engine. The algorithm encrypts the same sequence of symbols differently each time (and also gives different degrees of compression). This dynamic nature of the algorithm makes it immune to known plaintext attack, chosen plaintext attack and meet in the middle attack.

Once the engine has been running for a long time, which is the way it is intended to be used, and the number of occurring symbols is closer to the number of symbols in the alphabet, swapping occurs all over the alphabet range and symbol ordering cannot be found out. Thus, the security of the encrypted text resides in the following.

1) *The secret key:* In this case a brute force attack would require the number of attempts equal to the key entropy, which is a problem of the number of precision bits allowed for the key.
2) *The current state of the model:* This is more difficult to find by brute force, since the model of the arithmetic coder depends on all text that has been coded since the initialization of the model.

At any point, the cryptanalyst can approximate a value lying between the current values of H and L, both of which are 32 bits long. The strength, consequently, also depends on the 32 bit variables used to store symbol boundaries for 256 different symbols. In addition, the cryptanalyst has to approximate the next iterated value from the bitstream generator, in order to attempt to guess the swapping order, so that he can update his approximate model to account for the occurrence as well as consequent swaps of the characters. Since the bitstream generator is a result of multiple chaotic systems, information about the chaotic orbits for further approximation cannot be found out.

Apart from this, output from the engine is in the form of variable-sized words, which are fixed for any implementation, and individual bit output corresponding to inserted symbols cannot be determined. Statistical inferences cannot be correctly drawn from the output since swapped symbol spaces lead to unpredictable and incomputable errors/deviations in frequency. The model also defies attempts to steer it toward a favorable direction, as has been tried in [37]. We conclude our argument with a bitmap depicting the bit-output of repeated strings of a symbol, as shown in Fig. 9, which contains no discernible repetitive pattern. The first layer is the original string, while the second layer, which appears shorter due to compression, is the encoded string. In this case only one symbol is occurring and no exchanges take place.

VI. Conlusion

We have proposed a novel technique for: 1) secure encryption and 2) compression of data, which relies on a standard zeroth-order adaptive arithmetic coder for compression and makes the arithmetic coder's statistical model variable in nature using bitstream generated by the CCS PRBG. Chaos-based arithmetic coder and decoder have been designed and developed and the implementation details have been explained. Typical text files have been used to test our algorithm. It has been found that normal English text files are compressed to between 67.5% and 70.5%, the variations resulting from changing seeds values. This implies a loss of about 6% over a normal zeroth-order arithmetic coding model. Further, there was found no direct correlation between Δf_o and the degree of compression. The threshold parameter, t, has been found to adversely affect the degree of compression achieved. In another test, repeated strings of a symbol yielded encoded data which showed no repetition or underlying pattern. The nature of symbol swapping algorithm is such that no information about symbol ordering may be found out. Moreover, output from the engine is in the form of fixed-sized words, and individual bit output corresponding to inserted symbols cannot be determined. Statistical inferences cannot be correctly drawn from the output since swapped symbol spaces lead to unpredictable and incomputable errors/deviations in frequency. The model also defies attempts to steer it toward a favorable direction. The model has, thus, been shown to be largely secure and is not vulnerable to previous attacks against arithmetic coding.

The proposed algorithm is resistant to chosen plaintext attacks because of the following.

1) The model *dynamically* reorders the frequency of the input symbols according to the coupled chaotic system, and depends on all text that has been coded since the initialization of the model.
2) The model *dynamically* reorders the frequency of the input symbols according to the coupled chaotic system, and depends on all text that has been coded since the initialization of the model.
3) The output from the engine is in the form of variable-sized words and the individual bit output corresponding to inserted symbols cannot be determined.

Also, combination of the random behavior of conventional PRNGs and chaos-based RNGs in the CCS PRBG, may thwart cryptanalysis on both the fronts: chaos theory-based methods as well as conventional cryptanalysis. Recent fast implementations of the CCS PRBG as well as arithmetic coding result in a fast cipher with good security features.

REFERENCES

[1] B. Schneier, *Applied Cryptography*. New York: Wiley, 1996.

[2] C. Finnila, "Combining data compression and encryption," in *WESCON/94 "Idea/Microelectronics" Conf. Rec.*, Anaheim, CA, Sep., 27–29 1994, pp. 404–409.

[3] R. Brown and L. O. Chua, "Clarifying chaos: Examples and counterexamples," *Int. J. Bifurc. Chaos*, vol. 6, no. 2, pp. 219–249, 1996.

[4] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *Int. J. Bifurc. Chaos*, vol. 8, no. 6, pp. 1259–1284, 1998.

[5] L. Kocarev, G. Jakimoski, T. Stojanovski, and U. Parlitz, "From chaotic maps to encryption schemes," in *Proc. IEEE Symp. Circuits Syst.*. Monterey, CA, May–June, 31–3 1998, pp. 514–517.

[6] G. Alvarez, G. P. F. Monotoya, G. Pastor, and M. Romera, "Chaotic cryptosystems," in *Proc. IEEE 33rd Ann. Int. Carnahan Conf. Security Technol.*, Madrid, Spain, Oct. 4–8, 1999, pp. 332–338.

[7] F. Dachselt and W. Schwarz, "Chaos and cryptography," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 12, pp. 1498–1509, Dec. 2001.

[8] L. Kocarev, "Chaos-based cryptography: A brief overview," *IEEE Circuits Syst. Mag.*, vol. 1, no. 3, pp. 6–21, Mar. 2001.

[9] S. Li, X. Mou, and Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its application in stream-ciphers cryptography," in *Progress in Cryptology—INDOCRYPT 2001*. Chennai, India: Springer-Verlag, Dec. 16–20, 2001, pp. 316–329.

[10] R. Matthews, "On the derivation of a chaotic encryption algorithm," *Cryptologia*, vol. XIII, no. 1, pp. 29–42, 1989.

[11] M. BaptistaS, "Cryptography with chaos," *Phys. Lett. A*, vol. 240, no. 1–2, pp. 50–54, 1998.

[12] R. Bose and A. Banerjee, "Implementing symmetric cryptography using chaos functions," in *Proc. 7th Int. Conf. Advanced Commun. Comp. (ADCOM)*, Dec. 20–22, 1999, pp. 318–321.

[13] E. Alvarez, A. Fernández, P. García, J. Jiménez, and A. Marcano, "New approach to chaotic encryption," *Phys. Lett. A*, vol. 263, pp. 373–375, 1999.

[14] G. Jakimoski and L. Kocarev, "Chaos and cryptography: Block encryption ciphers based on chaotic maps," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 2, pp. 163–169, Feb. 2001.

[15] L. Kocarev and G. Jakimoski, "Logistic map as a block encryption algorithm," *Phys. Lett. A*, vol. 289, no. 4–5, pp. 199–206, 2001.

[16] K. W. Wong, "A fast chaotic cryptographic scheme with dynamic look-up table," *Phys. Lett. A*, vol. 298, no. 4, pp. 238–242, 2002.

[17] S. Li, X. Zheng, X. Mou, and Y. Cai, "Chaotic encryption scheme for real-time digital video," in *Proc. SPIE*, vol. 4666, 2002, pp. 149–160.

[18] D. D. Wheeler, "Problems with chaotic cryptosystems," *Cryptologia*, vol. XIII, no. 3, pp. 243–250, 1989.

[19] D. D. Wheeler and R. Matthews, "Supercomputer investigations of a chaotic encryption algorithm," *Cryptologia*, vol. XV, no. 2, pp. 140–151, 1991.

[20] E. Biham, "Cryptoanalysis of the chaoticmap cryptosystem suggested at EuroCrypt'91," *Adv. Cryptol.–EuroCrypt'91*, vol. 0547, pp. 532–534, Apr. 8–11, 1991.

[21] H. Zhou and X.-T. Ling, "Problems with the chaotic inverse system encryption approach," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 44, no. 3, pp. 268–271, Mar. 1997.

[22] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, "Cryptanalysis of a chaotic encryption system," *Phys. Lett. A*, vol. 276, pp. 191–196, 2000.

[23] G. Jakimoski and L. Kocarev, "Analysis of some recently proposed chaos-based encryption algorithms," *Phys. Lett. A*, vol. 291, no. 6, pp. 381–384, 2001.

[24] S. Li, X. Mou, and Y. Cai, "Improving security of a chaotic encryption approach," *Phys. Lett. A*, vol. 290, no. 3/4, pp. 127–133.

[25] S. Li, X. Mou, Y. Cai, Z. Ji, and J. Zhang, "On the security of a chaotic encryption scheme: Problems with computerized chaos in finite computing precision," *Comp. Phys. Commun.*, vol. 153, no. 1, pp. 52–58, 2003.

[26] S. Li, X. Mou, B. L. Yang, Z. Ji, and J. Zhang, "Problems with a probabilistic encryption scheme based on chaotic systems," *Int. J. Bifurc. Chaos*, no. 10, pp. 3063–3077, 2003.

[27] A. M. Frieze, R. Kannan, and J. C. Lagarias, "Linear congruential generators do not produce random sequences," in *Proc. 25th Ann. Symp. Found. Comp. Sci.*, Singer Island, FL, Oct. 24–26, 1984, pp. 480–484.

[28] J. Stern, "Secret linear congruential generators are not cryptographically secure," in *Proc. 25th Ann. Symp. Found. Comp. Sci.*, Los Angeles, CA, Oct. 12–14, 1987, pp. 421–426.

[29] C. Li, S. Li, G. Chen, G. Chen, and L. Hu, "Cryptanalysis of a new signal security system for multimedia data transmissions," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 8, pp. 1277–1288, 2005.

[30] L. Y. Deng and H. Xu, "A system of high-dimensional, efficient, long-cycle and portable uniform random number generators," *ACM Trans. Model. Comp. Simul.*, vol. 13, no. 4, pp. 299–309, 2003.

[31] P. L'Ecuyer and R. Touzin, "On the deng-lin random number generators and related methods," *Stat. Comput.*, vol. 14, no. 2004, pp. 5–9, 2004.

[32] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[33] P. G. Howard and J. S. Vitter, "Analysis of arithmetic coding for data compression," *Inf. Process. Manag.*, vol. 28, no. 6, pp. 749–763, 1992.

[34] P. G. Howard and J. S. Vitter, "Practical implementation of arithmetic coding," in *Image and Text Compression*, J. A. Storer, Ed. Norwell, MA: Kluwer Academic, 1992, pp. 85–112.

[35] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Trans. Inf. Syst.*, vol. 16, no. 3, pp. 256–294, 1998.

[36] H. A. Bergen and J. M. Hogan, "Data security in a fixed-model arithmetic coding compression algorithm," *Comp. Secur.*, vol. 11, pp. 445–461, 1992.

[37] H. A. Bergen and J. M. Hogan, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Comp. Secur.*, vol. 12, pp. 157–167, 1993.

[38] I. H. Witten and J. G. Cleary, "On the privacy afforded by adaptive text compression," *Comp. Secur.*, vol. 7, pp. 397–408, 1988.

[39] P. W. Moo and X. Wu, "Resynchronization properties of Arithmetic Coding," in *Proc. IEEE Data Compr. Conf.*, Snowbird, UT, Mar. 29–31, 1999, p. 540.

[40] X. Wu and P. W. Moo, "Joint image/video compression and encryption via high-order conditional entropy coding of wavelet coefficients," in *Proc. IEEE Int. Multimedia Comput. Syst.*, vol. 2, Florence, Italy, Jun. 7–11, 1999, pp. 908–912.

[41] S. Li, Q. Li, W. Li, X. Mou, and Y. Cai, "Statistical properties of digital piecewise-linear chaotic maps and their roles in cryptography and pseudorandom coding," in *Cryptography and Coding—8th IMA International Conference Proceeding*. Berlin, Germany: Springer-Verlag, 2001, pp. 205–221.

[42] S. Tao, W. Ruili, and Y. Yixun, "Perturbance-based algorithm to expand cycle length of chaotic key stream," *Electron. Lett.*, vol. 34, no. 9, pp. 873–874, 1998.

[43] S. Tao, W. Ruili, and Y. Yixun, "Clock-controlled chaotic keystream generators," *Electron. Lett.*, vol. 34, no. 20, pp. 1932–1934, 1998.

[44] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[45] T. Iokibe, M. Koyama, and M. Taniguchi, "A study of complexity of chaotic time series and prediction accuracy," in *Proc. IEEE Fuzzy Syst. Conf.*, Aug., 22–25 1999, pp. U-1004–U-1008.

[46] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Comput. Surveys*, vol. 19, no. 3, pp. 261–297, 1987.

[47] P. M. Fenwick, "A new data structure for cumulative frequency tables," *Softw. Pract. Exper.*, vol. 24, no. 3, pp. 327–336, 1994.

Ranjan Bose (M'98) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1992 and the M.S. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, Philadelphia, in 1993 and 1995, respectively.

He worked at Alliance Semiconductor Inc., San Jose, as a Senior Design Engineer from 1996 to 1997. Since November 1997, he has been with the Department of Electrical Engineering, Indian Institute of Technology, Delhi, where he is currently an Associate Professor. His current research interests lie in the areas of ultra-wideband (UWB) communications, coding theory and cryptography. He is the author of a book titled *Information Theory, Coding and Cryptography* (McGraw Hill, 2003).

Dr. Bose received the URSI Young Scientist award in 1999, the Humboldt Fellowship in July 2000, the Indian National Academy of Engineers (INAE) Young Engineers Award in 2003 and the AICTE Career Award for Young Teachers in 2004, and the BOYSCAST Fellowship in 2005.

Saumitr Pathak received the B.E. degree from Pantnagar University, Pantnagar, India, in 2002.

Since 2003, he has been a Research Associate in the Department of Electrical Engineering , the Indian Institute of Technology, New Delhi, India, on a project titled "Secure Communication using Chaos Theory."