Theory and Applications of Cellular Automata in Cryptography

S. Nandi, B. K. Kar, and P. Pal Chaudhuri

Abstract— **This paper deals with the theory and application of Cellular Automata (CA) for a class of block ciphers and stream ciphers. Based on CA state transitions certain fundamental transformations are defined which are block ciphering functions of the proposed enciphering scheme. These fundamental transformations are found to generate the simple (alternating) group of even permutations which in turn is a subgroup of the permutation group. These functions are implemented with a class of programmable cellular automata (PCA) built around rules 51, 153, and 195. Further, high quality pseudorandom pattern generators built around rule 90 and 150 programmable cellular automata with a rule selector (i.e., combining function) has been proposed as running key generators in stream ciphers. Both the schemes provide better security against different types of attacks. With a simple, regular, modular and cascadable structure of CA, hardware implementation of such schemes idealy suit for VLSI implementation.**

Index Terms—**Cryptography, block ciphers, cellular automata, simple group, even permutations, stream ciphers and key stream generator.**

I. Introduction

THE DESIGN style of digital logic has been significantly influenced by the VLSI technology. Designers always look for simple, regular, modular and cascadable logic circuit structure to realize a complex function. All these parameters are supported by the local neighborhood additive Cellular Automata (CA). Its applications in various fields have been proposed in [2]–[5], [15], [21], [22]. Recently, theory and application of CA as pseudo exhaustive pattern generators has been reported in [16]. Theory and applications for generation of error correcting code using CA is reported in [17]. Further, VLSI applications have been also proposed [23]–[25]. Application of CA in stream cipher cryptography was mentioned by Wolfram [26] with nonlinear CA rule. In the present paper, we establish the theory and application of additive CA as the basic data encryption hardware module.

With the ever increasing growth of data communication, the need for security and privacy has become a basic necessity. Cryptography is an essential requirement for communication privacy or concealment of data in a data bank. Encryption may be achieved by constructing two different types of ciphers—stream ciphers and block ciphers. A block cipher is the one in which a massage is broken into successive blocks and they are encrypted using single key or multiple keys. On the other hand, in a stream cipher the message is broken into successive bits or characters and then the string of characters is encrypted using a key stream. In the present paper schemes for a class of block ciphers and stream ciphers have been developed around the rugular structure of CA.

The block cipher operates on vectors of the N dimensional vector space V_N over GF(2). Certain transformations, to be called fundamental transformations in this paper, have been constructed using CA state transitions. These fundamental tranformations are the block ciphering functions of the proposed enciphering scheme. In the enciphering process, these transformations are applied on the clear text vectors resulting in encrypted vectors. Thus the block ciphering function is a randomly chosen function of V_N onto itself and it poses a difficult challenge to the intruders. The mathematical tool used in the present scheme is the permutation group S_{V_N} on the vector space V_N. It has been shown that the fundamental transformations form a well defined subgroup of S_{V_N}. This subgroup is the alternating group or the simple group A_{2^N} where A_{2^N} is the group of all even permutations on V_N. The fundamental transformations are built around additive CA rules having group properties, and are simple in nature. In this regard this scheme is applicable to encipherment of data stored in a computer, as well as to transmission of secret messages. One striking feature of this enciphering scheme lies in its good measure of strength. While another important characteristic is that the fundamental transformations are self inverses, resulting in that decipherment is carried out exactly in the same way as encipherment. A class of block ciphers arising in computer privacy has been studied in [6], [8]–[10], our present block cipher scheme being a particular case of this class.

The stream ciphers play an important role in cryptographic practices—both diplomatic and military—that protect communications in very high frequency domain. The central problem in stream cipher cryptography, however, is the difficulty of generating a long unpredictable sequence of binary signals from a short and random key. Unpredictable sequences are desirable in cryptography because it is impossible, given a reasonable segment of its signals and computer resources, to find out more about them. Pseudorandom bit generators have been widely used to construct these sequences [6], [4], [8]–[11], [13], [14]. Three basic requirements for *cryptographically secure* key stream generators are as follows.

Manuscript received February 5, 1993; revised July 8, 1993.

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur-721 302, India; e-mail: ppc@cse.iitkgp.ernet.in.

IEEE Log Number 9404360.

1) The period of the key stream must be large enough to accommodate the length of the transmitted message.
2) The output bits must be easy to generate.
3) The output bits must be hard to predict. That is, given a portion of the output sequence, the cryptanalyst should not be able to generate other bits forward or backward.

Application of CA in various fields has already been proposed in the literature [2], [15]. One of the applications project CA as pseudorandom pattern generators. Quality of randomness has been evaluated as per the criteria set by Knuth [18]. It has been established that the pattern generated by maximal length CA's meet all the criteria and the quality of randomness of the patterns generated by CA's is significantly better than that of Linear Feedback Shift Register (LFSR) based structures. Further enhancement of the quality of randomness can be embedded with the help of a Programmable CA (PCA) with dynamically alterable rule vector. The proposed key stream generator based on a PCA satisfies all the three essential yardsticks of "secure pseudorandom bit generators" noted above.

The proposed schemes for both block and stream ciphers can be found to be attractive in view of the following facts.

1) As per the work reported in [12], [19], DES-like function can be realized with the alternating group. The structure of PCA can be employed to generate the alternating group. This results in considerable saving of hardware compared to the existing schemes.
2) The programmability feature of PCA has enabled the realization of nonlinear enciphering function along with the generation of a large number of multiple keys rather than a fixed set key(s) usually employed.
3) For higher speed of operation, hardwired implementation of enciphering and deciphering schemes is a necessity. The schemes reported in the present work employ the regular, modular and cascadable structure of the three neighborhood CA that idealy suits for VLSI implementation. It can be operated at higher speed due to local interconnections.
4) Encipherment and decipherment proceed with the same protocols.
5) Measure of strength against intrusion can be found to be comparable, if not better, than the existing schemes.

II. Preliminaries

A. CA Preliminaries

CA is an array of sites (cells) where each site is in any one of the permissible states. At each discrete time step (clock cycle) the evolution of a site value depends on some rule (the combinational logic) which is a function of the present state of k of its neighbors for a k-neighborhood CA. Wolfram [1] has investigated cellular automata using empirical observations and simulations. For 2-state 3-neighborhood CA, the evolution of the ith cell can be represented as a function of the present states of $(i-1)$th, (i)th, and $(i+1)$th cells as: $x_i(t+1) = f\{x_{i-1}(t), x_i(t), x_{i+1}(t)\}$, where f represents the combinational logic.

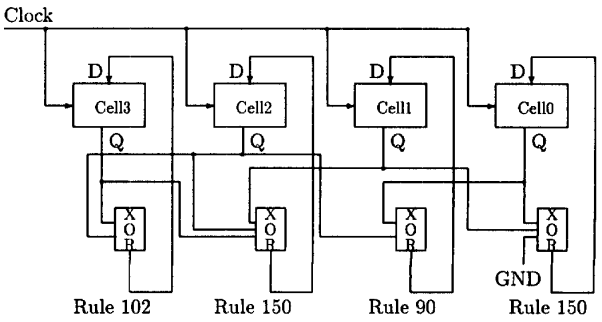


Fig. 1. A hybrid CA.

For a 2-state 3-neighborhood cellular automaton there are 2^3 distinct neighborhood configurations and 2^{2^3} distinct mappings from all these neighborhood configurations to the next state, each mapping representing a CA rule. The CA, characterized by a rule known as rule 90, specifies an evolution from neighborhood configuration to the next state as:

111	110	101	100	011	010	001	000	
0	1	0	1	1	0	1	0	Decimal 90

The corresponding combinational logic of rule 90 is

$$x_i(t+1) = x_{i+1}(t)\overline{x}_{i-1}(t) + \overline{x}_{i+1}(t)x_{i-1}(t) = x_{i-1}(t) \oplus x_{i+1}(t),$$

that is, the next state of ith cell depends on the present states of its left and right neighbors (Fig. 1). Similarly, the combinational logic for rule 150 is given by

$$x_i(t+1) = x_{i-1}(t) \oplus x_i(t) \oplus x_{i+1}(t),$$

that is, the next state of ith cell depends on the present states of its left and right neighbors and on its own present state (Fig. 1).

A CA characterized by EXOR and/or EXNOR dependence is called an additive CA. If in a CA the neighborhood dependence is EXOR, then it is called a noncomplemented CA and the corresponding rule is referred to as a noncomplemented rule. For neighborhood dependence of EXNOR (where there is an inversion of the modulo-2 logic), the CA is called a complemented CA. The corresponding rule involving the EXNOR function is called a complemented rule. In a complemented CA, single or multiple cells may employ a complemented rule with EXNOR function. There exist 16 additive rules which are:

Rule 0, 15, 51, 60, 85, 90, 102, 105, 150,
153, 165, 170, 195, 204, 240 & 255.

If in a CA the same rule applies to all cells, then the CA is called a **uniform CA**; otherwise the CA is called a **hybrid CA** (Fig. 1). There can be various boundary conditions; namely, **null** (where extreme cells are connected to logic '0'), **periodic** (extreme cells are adjacent), etc.

The following three complemented rules are employed for our block cipher scheme; the logic function for the corresponding noncomplement rules are also noted in Table I.

TABLE I
RULES USED IN BLOCK CHIPER SCHEME

complement			noncomplement	
rule	logic function	dependency	rule	logic function
195	$\overline{x_{i-1}(t) \oplus x_i(t)}$	left & self	60	$x_{i-1}(t) \oplus x_i(t)$
153	$\overline{x_i(t) \oplus x_{i+1}(t)}$	self & right	102	$x_i(t) \oplus x_{i+1}(t)$
51	$\overline{x_i(t)}$	self	204	$x_i(t)$

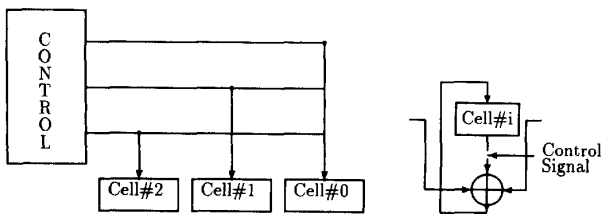


Fig. 2. A 3-cell Programmable CA structure and a PCA cell.

Complete characterization of additive CA based on matrix algebraic tools has been reported in [3], [2], [16]. The characteristic matrix T of a CA and the characteristic polynomial are illustrated with the following example. The next state $f_{t+1}(x)$ of an additive CA is given by $f_{t+1}(x) = T \times f_t(x)$, where $f_t(x)$ is the current state, t is the time step.

Example 1: A four cell null boundary hybrid CA (Fig. 1), with the rule vector $\langle$ 102, 150, 90, 150 $\rangle$ applied from left to right, may be characterized by the following characteristic matrix

$$T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

$$T + \lambda I = \begin{bmatrix} 1+\lambda & 1 & 0 & 0 \\ 1 & 1+\lambda & 1 & 0 \\ 0 & 1 & \lambda & 1 \\ 0 & 0 & 1 & 1+\lambda \end{bmatrix}.$$

The characteristic polynomial is $\lambda^4 + \lambda^3 + 1$; $T^{15} = I$.

If the characteristic polynomial of a CA is primitive, then it is referred to as a maximal length CA. Such an L cell CA generates all the $2^L - 1$ states in successive cycles, excluding the all zero state. If all the states of the CA form a single or multiple cycles, then it is referred to as a group CA. The following Lemmas are required for the mathematical foundation of our schemes.

Lemma 1 [2]: A CA is a group CA iff $T^p = I$, I being the identity matrix and p is positive integer.

Lemma 2 [2]: Let $\overline{T}^p$ denoting application of the complemented rule $\overline{T}$ for p successive cycles, then

$$\overline{T}^p f(x) = [I + T + T^2 + \cdots + T^{p-1}][F(x)] + [T^p][f(x)]$$

where T is the characteristic matrix of the corresponding noncomplemented rule vector and $[F(x)]$ is an L-dimensional vector (L = number of cells) responsible for inversion after EXORing. $F(x)$ has '1' entries (i.e., nonzero entries) for CA cell positions where EXNOR function is employed.

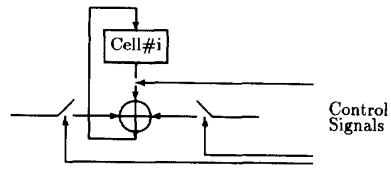


Fig. 3. A more general PCA cell with three control lines.

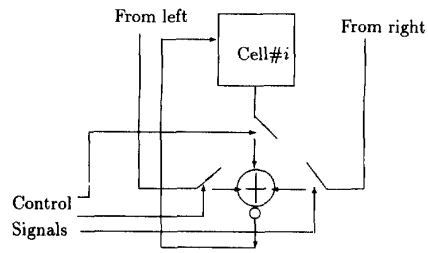


Fig. 4. A PCA cell with EXNOR, rule.

Lemma 3 [2]: The complement of a group CA is also group CA.

Lemma 4 [4]: CA rules 60, 102 and 204 form groups for all lengths (l) with a group order $O(G) = n = 2^a; a = 0, 1, 2, \cdots; n \geq l > n/2$.

Programmable CA (PCA): Positional representations of *rule* 90 and *rule* 150 show that their neighborhood dependence differ in only one position, viz., on the cell itself. Therefore, by allowing a single control line per cell (Fig. 2), one can apply both *rule* 90 and *rule* 150 on the same cell at different time steps. Thereby, an L cell CA structure can be used for implementing 2^L CA configurations. Realizing different CA configurations on the same structure can be achieved using a control logic to control the appropriate switches and a control program, stored in a ROM, can be employed to activate the control. The 1(0) state of the ith bit of a ROM word closes (opens) the switch that controls the ith cell. Such a structure is referred to as a *Programmable CA*. Fig. 2 shows a programmable CA with simple control structure—allowing one control input per cell that configures the rule, applied to that cell, either to rule 90 or rule 150. The L bit control word for an L cell PCA has 1(0) on ith cell if the rule 150(90) is applied to the ith cell. For example, the control word $\langle 0110 \rangle$ for a four cell PCA leads to the 1st and 4th cells to be configured with rule 90, while the 2nd and 3rd cells are configured with rule 150. Such a structure has been used in our stream ciphering scheme.

By allowing more complex control one can introduce immense flexibility to this programmable structure. A more general 3-neighborhood programmable CA cell is shown in Fig. 3. Using such a cell structure for all the cells, all possible additive noncomplemented rule combinations can be achieved to realize any hybrid additive CA. Such an L cell PCA can implement 2^{3^L} number of different CA configurations. A 3-neighborhood programmable CA cell with complemented additive rule is shown in Fig. 4. While Fig. 5(a) shows a programmable CA with lesser number of control switches, in which rule 51, 153 or 195 can be realized. Such a structure

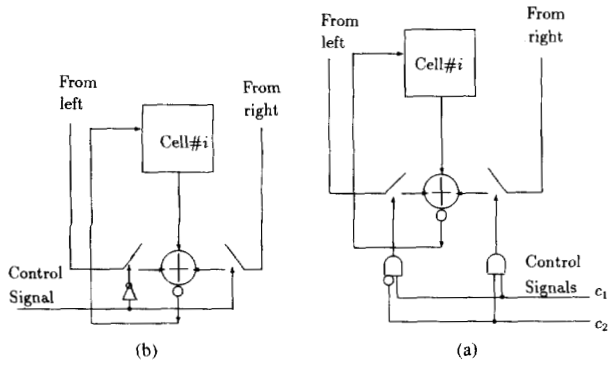


Fig. 5. A PCA cell with EXNOR rule and modified control lines.

is employed for block ciphers. The value of control switches (c_1, c_2) corresponds to

c_1c_2	rule applied
00 or 01	51
10	195
11	153.

Fig. 5(b) shows a circuit with reduced control logic where either rule 153 or 195 can be employed in a cell.

B. Permutation Groups

A permutation π of a finite set $S = \{x_1, x_2, \cdots, x_n\}$ is defined to be an injection $\pi : S \longrightarrow S$. It can be noted that all possible permutations on S form a noncommutative group G_n of order $n!$ under the operation of permutation multiplication.

An important feature of a permutation is its cyclic structure. By a cycle $(y_1, y_2, \cdots, y_r)$ of length r, we mean permutation π in S such that $\pi(y_j) = y_{j+1}$, $j = 1, \quad 2, \cdots, r-1$, and $\pi(y_r) = y_1$, while leaving all other elements of S fixed.

It is easy to see that every permutation may be uniquely expressed as a product of disjoint cycles. By a transposition we shall mean a cycle of length two. An even permutation is the one that is expressible as a product of an even number of transpositions; otherwise a permutation is called an odd permutation. It is a well known result that all even permutations on S form a normal subgroup of G_n, which is the alternating group A_n of degree n and order $(n!)/2$. In fact A_n is the only normal subgroup of index two of G_n—such a group is called a simple group.

III. Permutation Representation of CA States

Let us consider a nonmaximal length CA in which there are r cycles with c_i number of states on the ith cycle. Such a CA has a cyclic group representation of the form,$G = \{R, R^2, R^3, \cdots, R^x\}$; where $R^x = I$, the identity.

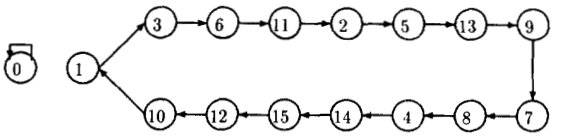


Fig. 6. The state transition diagram of a maximal length group CA.

In this representation R is the generator of G, and $x = \text{lcm}(c_i,\ i = 1, 2, \cdots, r)$; where lcm means lowest common multiple. Such a group may be called a rule group. This is equivalent to the fact that, a CA gives rise to a cyclic group if and only if the mapping corresponding to the state transition graph is a permutation. The reason behind this is the fact that, in case of a CA forming a rule group, each state in its transition graph has exactly one predecessor i.e., the transition mapping is a bijection which obviously has a permutation representation.

Example 2: Consider the 4 bit hybrid CA ⟨90, 150, 90, 150⟩ under the null boundary condition. It's state transition graph is shown in Fig. 6.

It is easy to see that the above CA forms the cyclic group

$$G = \{R, R^2, R^3, \cdots, R^{15} = I\}$$

While its state transition graph may be represented by the permutation found at the bottom of the page. It may also be stated that in case of a CA forming a rule group of order x and having the permutation representation π, one can have: $\pi^x = (0)(1)(2)\cdots(2^n - 1) = I$; n being the length of the CA.

A. Permutation Representation of CA Having Equal Cycles of Even Length

Lemma 4 (introduced in section 2) provides the CA rules that generate cycles of length $2^a, a = 0, 1, 2, \cdots$. The following Lemma establishes the corresponding results for uniform and hybrid CA's with complemented rules 51, 153, and 195. The corresponding noncomplemented rules are 204, 102 and 60.

Lemma 5: Complemented CA rules 195, 153 and 51 form groups for all lengths with group order $O(G) = m = 2^a$; $(a = 1, 2, 3, \cdots)$.

Proof: Consider a CA with rule R and characteristic matrix T, where R is a combination of the rules 60, 102, and 204. Then, as per Lemma 2, the corresponding complemented CA, with characteristic matrix $\overline{T}$, may be expressed as:

$$\overline{T}^m f(x) = [I + T + T^2 + \cdots + T^{m-1}][F(x)] + [T^m][f(x)]. \quad (1)$$

The fact that R is a group CA rule implies that $T^n = I$ for n as some integral power of 2 (Lemma 4). As per Lemma 3, complement of a group CA is also a group CA. So,

$$\overline{T}^m f(x) = f(x), \quad (2)$$

$$\pi = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 3 & 5 & 6 & 14 & 13 & 11 & 8 & 4 & 7 & 1 & 2 & 10 & 9 & 15 & 12 \end{bmatrix}$$

TABLE II
CA HAVING EQUAL CYCLES OF EVEN LENGTH

number of CA cells	Rule vector	generated cycles	
		number	length
4	51,51,51,51	8	2
4	153,153,153,153	2	8
4	195, 195,195,195	2	8
8	51,51,51,$\cdots$,51	128	2
8	153,153,153,$\cdots$,153	16	16
8	195,195,195,$\cdots$,195	16	16

where m is the cycle length of the complemented CA. From (1) and (2),

$$[T^m + I][f(x)] = [I + T + T^2 + \cdots + T^{m-1}][F(x)]$$
$$\Rightarrow [T + I][I + T + T^2 + \cdots + T^{m-1}][f(x)]$$
$$= [I + T + T^2 + \cdots + T^{m-1}][F(x)].$$

Assume $[I + T + T^2 + \cdots + T^{m-1}] \neq 0$, consequently

$$[T + I][f(x)] = [F(x)]. \quad (3)$$

If the CA under consideration consists of L number of cells, then (3) is a system of L linear equations, and the condition for its solution to exist is

$$\text{rank}[T + I] = \text{rank}[T + IF(x)].$$

In the case of R, being any combination of rules 60, 102 and 204, it can be directly shown that rank $[T + I] < L$, owing to fact that one row of matrix $T + I$ is null in such a case. Also, since each entry of $F(x)$ is 1 (as in the case of all complemented rules), it follows that

$$\text{rank}[T + I] \neq \text{rank}[T + IF(x)].$$

This is a contradiction and, hence, it follows that

$$1 + T + T^2 + \cdots + T^{m-1} = 0 \quad (4)$$
$$\Rightarrow \overline{T}^m[f(x)] = T^m[f(x)] = f(x) \quad (5)$$
$$\Rightarrow T^m = I.$$

Let $m = bn$, where b is nonzero positive integer. For $b = 2$,

$$\begin{aligned} & I + T + T^2 + \cdots + T^{m-1} \quad (\text{as } m = 2n) \\ = \; & I + T + T^2 + \cdots + T^{n-1} + T^n \\ & + T^{n+1} + T^{n+2} + \cdots + T^{2n-1} \\ = \; & [I + T + T^2 + \cdots + T^{n-1}] \\ & + [I + T + T^2 + \cdots + T^{n-1}] \quad (\text{as } T^n = I) \\ = \; & 0 \quad (\text{since modulo-2 summation is involved}). \end{aligned}$$

So, the relation (4) always satisfies for $b = 2$. For particular values of T, relation (4) may hold for $b = 1$. Hence, the value of m is either n or $2n$.

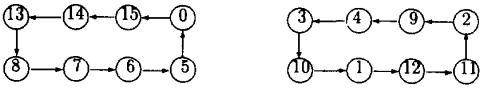


Fig. 7. The state transition diagram of a nonmaximal length group CA.

Now we need to show that m is a nonzero positive integral power of 2. As per Lemma 4 in Section 2, n is of the form 2^a, $(a = 0, 1, 2, \cdots)$. We consider the following two cases.

Case 1: for $n = 2^0 = 1$

$$\Rightarrow T = I$$
$$\Rightarrow I + T = 0 \quad (6)$$

Considering equations (4) and (6) we arrive at the conclusion that $m = 2$ for $n = 1$.

Case 2: for $n = 2^a$, $(a = 1, 2, 3, \cdots)$;
we know that m is either n or $2n$.
So m is also a nonzero positive integral power of 2. □

Theorem 1: If a null boundary uniform or hybrid CA configured with rules 51, 153 and 195 is a group CA, then its state transition diagram consists of equal cycles of even length.

Proof: From Lemma 5, it can be seen that group CA, under different configurations of rules 51, 153, and 195, generate cycles of even length m (positive integral power of 2). Now we have to prove that factors of m can not be a cycle lengths. Assume that the group CA has a cycle of length m_i [where m_i is a factor of m]. Then it must satisfy the following equations:

$$I + T + T^2 + \cdots + T^{m_i - 1} = 0$$
$$\text{and} \quad \overline{T}^{m_i}[f(x)] = T^{m_i}[f(x)] = f(x).$$

This implies that m_i is the group order of all cycle lengths of the group CA, suggesting that m_i is equal to m, i.e., all cycles are equal in length.

Hence, the theorem. □

For the sake of illustration, cycle structures of a few null boundary CAs with rules 51, 153, and 195 are tabulated in Table II.

In case of periodic boundary conditions, the only rule available forming equal cycles of even length is rule 51. This phenomenon is true for CA of any length.

Now considering the CA ⟨153, 153, 153, 153⟩, under null boundary conditions, we obtain cycles as shown in Fig. 7.

This CA has the permutation representation found at the bottom of the page, where π^4, being a product of even number of transpositions, is an even permutation.

$$\pi = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 12 & 9 & 10 & 3 & 0 & 5 & 6 & 7 & 4 & 1 & 2 & 11 & 8 & 13 & 14 \end{bmatrix}$$
$$\Rightarrow \pi^2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 14 & 11 & 4 & 1 & 10 & 15 & 0 & 5 & 6 & 3 & 12 & 9 & 2 & 7 & 8 & 13 \end{bmatrix}$$
$$\Rightarrow \pi^4 = (0,8)(1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15)$$

TABLE III
EVEN NUMBER OF EVEN LENGTH CYCLES OF CA FOR $L = 8$ AND 16

Rule applied to cells	Number of CA configurations generating even number of even length cycles	
	8 cell CA having 8 length cycles	16 cell CA having 16 length cycles
51, 153, 195	691	493521
51, 153 (or 195)	184	47190
153, 195	7	523

In general, CA's having equal cycles of even length each, may be expressed as some even power of an even permutation. This phenomenon has been extensively used in our present encryption scheme. Table III shows a number of L cell CA configurations, each generating cycles of length 8 or 16.

IV. DEFINITION OF FUNDAMENTAL TRANSFORMATIONS

To encipher blocks of binary data we shall define certain transformation functions and call them fundamental transformations. These are fundamental in the sense that any enciphering function will be constructed using them. The fundamental transformations are built up using the following combinational logic representing CA rules 195, 153, and 51:

$$x_i(t) = \overline{x_{i-1}(t-1) \oplus x_i(t-1)} \quad (7)$$

$$x_i(t) = \overline{x_i(t-1) \oplus x_{i+1}(t-1)} \quad (8)$$

$$x_i(t) = \overline{x_i(t-1)}. \quad (9)$$

In the enciphering process, each bit of a block of message is input to a CA cell. Each cell is updated according to some transformations which are depicted in (7)–(9).

Each of these rules is a group rule and thus the corresponding CA exhibit rule group properties. It can also be observed that such a CA forms equal cycles of even length each—a phenomenon which is a basic requirement of our enciphering scheme. The cycle length distribution of this class of CA's has been discussed in Section III.

If $2r$ is the cycle length of uniform or hybrid CA with a rule R (i.e., a combination of 51, 153, and 195), then a fundamental transformation using the R rule CA is defined by $s = \pi^r$, where π is a permutation representation of the R rule CA.

It is easy to see that $\pi^{2r} = I$ (the identity permutation). So, s is an involution (i.e., self inverse transformation). It is also a transposition.

For example, the fundamental transformations s, y, z built up on 4-bit uniform null boundary CA using the aforementioned rules can be represented as follows:

$$s = \pi_1^4 \quad (10)$$

$$y = \pi_2^4 \quad (11)$$

$$z = \pi_3 \quad (12)$$

where the permutation representations of π_1, π_2, and π_3 are in (13)–(15), found at the bottom of the page. In these cases $\pi_1^8 = \pi_2^8 = \pi_3^2 = I$ (the identity permutation); resulting in involutions s, y, z which are also transpositions.

One can easily observe that:

$$s = (0,1)(2,3)(4,5)(6,7)(8,9)(10,11)(12,13)(14,15);$$
$$y = (0,8)(1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15);$$
$$z = (0,15)(1,14)(2,13)(3,12)(4,11)(5,10)(6,9)(7,8).$$

Each being a product of even number of transpositions. The fundamental transformations may be regarded as even permutations.

A. Fundamental Transformations Generate a Group

This subsection establishes the promising result that the fundamental transformations, constructed earlier, generate the simple group or alternating group A_{2^N} of even permutations, which in turn is a subgroup of the permutation group S_{2^N}. To achieve this we have the following results.

Theorem 2: A_{2^N} cannot be generated by a single basic transformation.

Proof: It is quite easy to observe that if A_{2^N} is generated by a single basic transformation g (say), then $A_{2^N} = \{g, \quad g^2 = I\}$ which is in contradiction to the fact that A_{2^N} contains as many as $(2^N)!/2$ elements. □

This obviously leads to the following corollary.

Corollary 1: If n is the number of basic transformations required to generate A_{2^N}, then $2 \leq n < 2^N - 1$.

Theorem 3: A_{2^N} is generated by 3-cycles.

Theorem 4: If a normal subgroup of A_{2^N} contains a 3-cycle then it must be the whole of A_{2^N}.

Theorem 3 and 4 are well established results, the proofs of which may be found in any standard text of abstract algebra, such as [7].

We illustrate our enciphering scheme for 8 bit block with five fundamental transformations. These transformations are

$$\pi_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 14 & 12 & 13 & 9 & 8 & 10 & 11 & 3 & 2 & 0 & 1 & 5 & 4 & 6 & 7 \end{bmatrix} \quad (13)$$

$$\pi_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 12 & 9 & 10 & 3 & 0 & 5 & 6 & 7 & 4 & 1 & 2 & 11 & 8 & 13 & 14 \end{bmatrix} \quad (14)$$

$$\pi_3 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}. \quad (15)$$

constructed using the following 8 bit null boundary CA with rules 153, 195, and 51. Each CA configuration generates cycles of length 8:

$$\langle 153, 153, 153, 153, 51, 51, 51, 51\rangle$$
$$\langle 195, 195, 195, 195, 51, 51, 51, 51\rangle$$
$$\langle 51, 51, 153, 153, 153, 153, 51, 51\rangle$$
$$\langle 51, 51, 195, 195, 195, 195, 51, 51\rangle$$
$$\langle 51, 153, 153, 153, 153, 153, 153, 51\rangle.$$

In order to reduce circuit complexity, only five such configurations with either rule 153 (or rule 195) along with rule 51 are chosen out of 184 available combinations. Each of these 184 hybrid CA's of 8 bit each generate equal cycles of even length. One is free to take any number of such fundamental transformations to enhance invulnerability of the scheme. Complexity analysis of this scheme against different types of attacks is noted in Section VII.

V. PCA Based Block Cipher Scheme

Let us recall that the N dimensional vector space V_N consists of all of our messages, each N bit long. In our present enciphering strategy, whole message is divided into a number of sub-blocks. Each sub-block will contain N bits if total message length is a multiple of N, else the leftmost block will be filled with zeros. Let us assume $N = 8$.

The space of our enciphering functions is S_{2^8} with cardinality $2^8!$ which is a considerably large number. Let $E \in S_{2^8}$ be an enciphering function and $M \in V_N$ be a message to be encrypted. Then our scheme yields the encrypted message $C = EM$. The enciphering function consists of q fundamental transformations. Each of the fundamental transformations is derived from different rules. By permutation of q fundamental transformations we can generate $q!$ enciphering functions.

As a specific example, let $E = syzuv$, with $q = 5$. Thus $C = EM$. In the decryption process, one obtains:

$$\begin{aligned} M &= E^{-1}C \\ &= (syzuv)^{-1}C \\ &= (v^{-1}u^{-1}z^{-1}y^{-1}s^{-1})C \\ &= (vuzys)C. \end{aligned}$$

Hence, to decrypt a message, the fundamental transformations are applied on the encrypted message in reverse order.

The fundamental transformations constructed taking into account the aforesaid considerations are:

$$s = \pi_1^4, \quad y = \pi_2^4, \quad z = \pi_3^4, \quad u = \pi_4^4, \quad v = \pi_5^4.$$

Here, $\pi_i^8 = I$; $i = 1, 2, 3, 4, 5$; implying that s, y, z, u, v are involutions.

The above scheme may be implemented through a circuit whose block diagram is presented in Fig. 8.

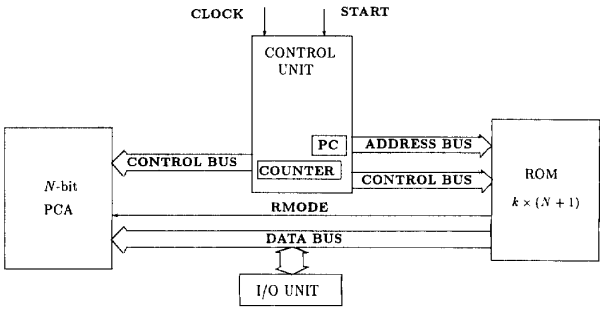


Fig. 8. Block diagram of the circuit for block cipher scheme.

A. Number of Enciphering Functions

In our block cipher scheme one message block is enciphered by one enciphering function. The deciphering, as noted earlier, is achieved by applying the same enciphering functions, with the fundamental transformations for each function being applied in the reverse order. Let p be the number of enciphering functions used in the proposed scheme, which are noted as $\{E_0, E_1, E_2, \cdots, E_{p-1}\}$, while for deciphering it is $\{E_0^{-1}, E_1^{-1}, E_2^{-1}, \cdots, E_{p-1}^{-1}\}$. In the block cipher process, the first block of message is enciphered with E_0, the second block with E_1, the third block with E_2, and so on. This process repeats until enciphering of all the message blocks is complete. The enciphering function changes in every message block i.e., if ith block is enciphered with E_i, then the next block enciphering function is $E_{(i+1)\mathbf{mod}_p}$. After every enciphering operation, the output of PCA is taken as the encrypted message block. At the receiving end, the deciphering process is analogously applied with $\{E_0^{-1}, E_1^{-1}, E_2^{-1}, \cdots, E_{p-1}^{-1}\}$. These p enciphering functions are constructed using permutation, of q fundamental transformations (i.e., each enciphering function consists of q fundamental transformations). The definition of fundamental transformation is given in the Section IV.

The fundamental transformations can be realized by PCA as follows. Load PCA with a state $f_t(x)$. Configure PCA with complemented rules 51, 153 and 195 as discussed in Section III-A. Then run the PCA for r cycles, i.e., $\overline{T}^r f_t(x) = f_{t+r}(x)$ [where $\overline{T}^{2r} f_t(x) = f_t(x)$].

The equivalent permutation representation (i.e., the fundamental transformation) of this is π^r [where $\pi^{2r} = I$, the identity permutation].

A fundamental transformation is realized with a CA configuration. The control signals corresponding to a CA configuration are stored in a ROM word. For an N-bit PCA having v control signals per CA cell, there are Nv-bits in a ROM word. An enciphering function is realized with q fundamental transformations. Therefore, q consecutive ROM address spaces are required to store one enciphering function. Hence, $p \times q$ ROM address spaces are sufficient to store p enciphering functions.

Now, for a given value of k (the ROM address space) and q (the number of fundamental transformations), the number of enciphering functions used in the block cipher scheme can be computed as follows.

— The number of enciphering functions that can be formed with q fundamental transformations is $q!$.
— The number of enciphering functions possible to store in ROM is $\lceil k/q \rceil$.
— So the number of enciphering functions (p) used by the scheme is $\min\{\lceil k/q \rceil, q!\}$.

The sequential steps for encryption/decryption are noted in the Algorithm 1.

Description of the Circuit

PCA (Programmable Cellular Automata): It is an N-bit null boundary, uniform or hybrid CA loaded with the rules 51, 195, and 153. The rules 153 and 195 do not appear simultaneously in the same CA configuration. The control structure of such a PCA cell is shown in Fig. 5(a). Selection of this type of rule configuration reduces the circuit complexity. The control signals for a CA configuration are stored in the ROM and loaded into the PCA via the DATA BUS. One bit signal RMODE is 0(1), indicating that the data bus contains rule 195(153) and 51. RMODE represents the c_2 input for all cells in the PCA. The control input c_1 for each of the cells is derived from the corresponding bit position of the ROM word. Thus for an N bit CA, the ROM word size is $N+1$ bit.

Control Unit: It consists of several counters to generate different types of control signals for PCA and ROM. These counters are as follows.

C_r: counts the number of clock cycles 'r' the PCA will run to realize a fundamental transformations.

C_q: counts the number of fundamental transformation 'q' in an enciphering function.

Program Counter (PC) is of $\lceil \log_2 k \rceil$-bit length modulo $q \times p$ counter and is used to store address of the ROM where next PCA rule configuration control word is located.

The control sequences of the circuit are described in the Algorithm 1 noted below.

ROM (Read Only Memory): It is of size $k \times (N+1)$, which can store rule configuration control word of PCA, with the $(N+1)$th-bit being used to control the RMODE control signal line.

I/O UNIT: It is an input / output unit for data transfer between PCA and outside world.

Only two external signals are required to operate the whole circuit, i.e., CLOCK for running the circuit and START for reset and start the circuit.

Algorithm 1:

Input: *For encryption (decryption) the input is plaintext (ciphertext).*

Step 1: *Reset all counters in control unit.*

Step 2: *While (not end of plaintext (ciphertext)) execute steps 3 to 6.*

Step 3: *Load PCA with one byte plaintex (ciphertext) from I/O UNIT and set $C_q = 0$.*

Step 4: *Load rule configuration control word from the ROM into the PCA and set $C_r = 0$.*

Step 5: *Run PCA for r-cycles.*

Step 6: *Increment C_q and PC by 1. If $(C_q = q)$ then send one byte ciphertext (plaintext) to I/O UNIT; otherwise go to step 4.*

VI. Stream Cipher Strategy

In stream ciphers pseudorandom pattern generators are widely used to generate the key stream.

A. Key Stream Generators

Many key stream generators are based on combining two or more generators (i.e., LFSR's) by using nonlinear functions [11], [13], [14]. It is already established that maximum length CA's generate patterns having high quality of pseudorandomness [20]. Using CA properties we have proposed two types of key stream generators-(1) PCA with ROM, and (2) Two stage PCA. Fig. 2 shows a 90 / 150 PCA cell used in the key stream generators.

PCA With ROM as a Key Stream Generator: Let L be the number of cells in the PCA and w be the number of maximal length CA's with rule 90 and rule 150. Assume that l maximal length CA's are chosen out of w maximal length CA's. These rules are noted as $\{R_0, R_1, R_2, \cdots, R_{l-1}\}$. The rule configuration control word corresponding to a rule R_i is stored in a ROM word. Initially the PCA is configured with rule R_0 and loaded with a non zero seed. With this configuration the PCA runs one clock cycle. Then it is reconfigured with the next rule (i.e., R_1) and runs another cycle. This process repeats until CLOCK signal to PCA is made inactive. The rule configuration of PCA changes after every run, i.e., if in the ith run rule configuration is R_i, then in the next run, rule is $R_{(i+1) \bmod l}$. After each clock cycle, the output of PCA is taken as a pseudorandom pattern.

Now our objective is to show that this output sequence is a pseudorandom pattern sequence. The following Theorem provides the background.

Theorem 5 [2], [20]: If the characteristic polynomial of a CA is primitive then it generates pseudorandom pattern.

Corollary 2: A PCA built with maximal length CA configurations generate pseudorandom patterns.

Proof: All the maximal length CA's generate pseudorandom sequences, individually. The sequence generated by the PCA can be taken as a set of subsequences generated by a particular maximal length CA. As the subsequences are pseudorandom in nature so the overall sequence is also pseudorandom in nature. □

The above key stream generator scheme may be implemented through a circuit whose block diagram is presented in the Fig. 9.

Description of the Circuit:

PCA (Programmable CA): It is an L-bit null boundary, uniform or hybrid CA configured with the rule 90 and 150. The control signals corresponding to a CA configuration are stored in the ROM and loaded into the PCA via the DATA BUS.

ROM (Read Only Memory): It is of size $l \times L$ (l words, each of L bits), and it stores the control signals for the PCA.

Control Unit: It consists of several counters to generate different types of control signals for PCA and ROM. The control sequences of the circuit are described in the Algorithm 2 below. Program Counter (PC) is $(\log_2 l)$-bit (i.e., l is power of 2) up counter and is used to store address of the ROM where next PCA rule is present.

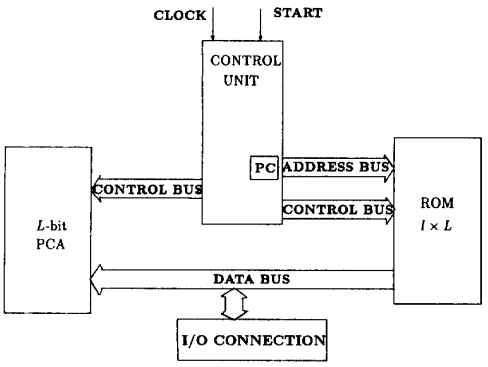


Fig. 9. PCA based pseudorandom pattern generator.

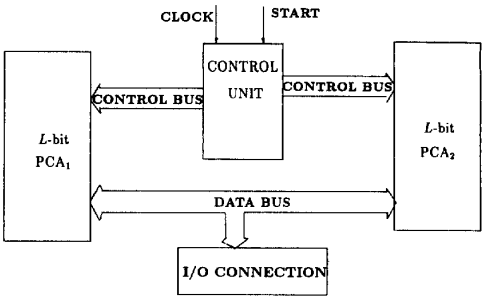


Fig. 10. Two stage PCA based pseudorandompattern generator.

I/O Connection: It is an input/output unit for data transfer between PCA and outside world.

Only two external signals are required to operate the whole circuit, i.e., CLOCK for running the circuit and START for reset and start the circuit.

Details of the Experimental PCA Chip: The PCA is the major block in the designs. In view of this, for building the prototype system, we have designed, fabricated and tested 16 bit and 32 bit PCA chips. The chips have been fabricated with a 3 micron gate array library at ITI (Indian Telephone Industries). Details of the 32 bit PCA chip with 90 and 150 rules are : 1) 64 pin DIP, 2) area equivalent 1284 two input NAND gates, 3) power -316 mW, (iv) current -57 mA, and (v) operating clock speed of 10 Mz.

Algorithm 2:

Step 1: *Reset all counters in the control unit.*

Step 2: *Load PCA with L-bit initial seed from I/O connection.*

Step 3: *Read the ROM control word and configure the PCA.*

Step 4: *Run PCA for one cycle.*

Step 5: *Read pseudorandom pattern from I/O connections and increment PC by* 1.

Step 6: *If (CLOCK active) then go to step* 3.

Step 7: *Stop.*

In the above scheme, the PCA configured by a rule (stored in the ROM) is assumed to run one cycle only. By using some extra ROM bits & additional control, we can specify the number of cycles the PCA should run for each configuration. Such modification can substantially enhance the quality of encipher.

Two Stage PCA as a Key Stream Generator: In the ROM based key stream generator, the main disadvantage is the increased area overhead with lower speed of operation due to use of ROM for storage of control signals. This can be avoided by replacing the ROM with another maximal length 90/150 rule PCA (i.e., PCA_2) as shown in the Fig. 10. A single chip can be fabricated with PCA_1, PCA_2 and the CONTROL UNIT. The control signals (R) to configure PCA_2 and the input seed (I_2) for PCA_2 can be concatenated to the input seed (I_1) of PCA_1 to form the key (i.e., $\langle R, I_2, I_1 \rangle$) for the key stream generator. The PCA_2 generates the control signals to configure PCA_1.

Description of the Circuit:

PCA_1(Programmable CA_1) It is an L-bit null boundary, uniform or hybrid CA loaded with the rule 90 and 150. The control signals to configure PCA_1 are loaded from the output of PCA_2 via the DATA BUS.

PCA_2(Programmable CA_2) It is an L-bit null boundary, uniform or hybrid CA configured with the rule 90 and 150. Rule (R) is part of the key and it is loaded into the PCA_2 via the DATA BUS.

Control Unit: It consists of several counters to generate different control signals for PCA_1 and PCA_2. The control sequences of the circuit are described in the Algorithm 3 below.

I/O Connection: It is an input / output unit for data transfer between PCA_1, PCA_2 and outside world.

Algorithm 3:

Step 1: *Reset all counters in the control unit.*

Step 2: *Configure PCA_2 with the control signals (R) from the I/O connections.*

Step 3: *Load PCA_1 and PCA_2 with initial seed I_1 and I_2.*

Step 4: *Run PCA_2 for one cycle.*

Step 5: *Configure PCA_1 with the control signals from the output of PCA_2.*

Step 6: *Run PCA_1 for one cycle.*

Step 7: *Output pseudorandom pattern from I / O connections (i.e. output of PCA_1).*

Step 8: *If (CLOCK active) then go to step* 4.

Step 9: *Stop.*

The enciphering process using this type of generator fails if PCA_1 goes to all zero graveyard state [2]. Analogous to modified LFSR design, with some extra logic it is possible to design the PCA_1 to have a transition out of all-zero state. On the other hand, it is necessary to avoid a situation where PCA_2 enters in a graveyard state resulting in PCA_1 being configured with the same rule all through out. So, the user of the scheme must avoid such a key combination from the simulation study.

B. PCA Based Stream Cipher Scheme

Fig. 11 depicts the class of proposed stream ciphers (where $L = 16$). The output of the message source is regarded as a stationary random sequence. Each component takes values

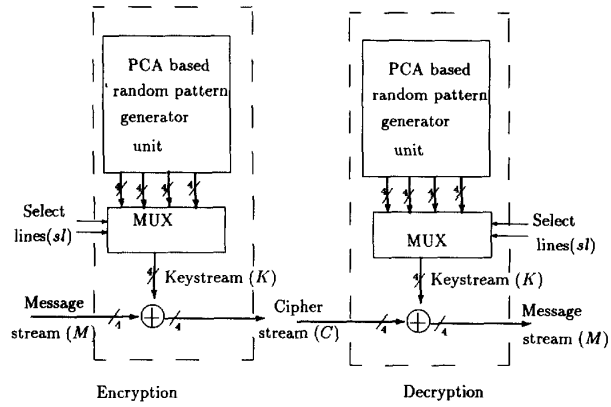


Fig. 11. Proposed stream cipher scheme.

from a finite set S_M. In this encipher the incoming message sequence is divided into message words of 4 bit in length.

The key source output is regarded as a random variable K taking equiprobable values in a set S_K with 2^L elements, where L is the length of PCA as described in the previous section. From the L bit key the proposed method takes any consecutive 4-bits for encryption. The key stream output and the message source output are statistically independent.

An encipher produces the ith bit ($i = 1, 2, 3, 4$) of cryptogram word C from the ith bit of the message word M and the ith bit of the key word K by using EXOR function.

$$C_i = M_i \oplus K_i.$$

This is a mapping from $S_M \times S_K$ to S_C, where S_C is the set of cryptogram word.

The deciphering strategy follows the same scheme where the EXOR gates produce message text taking cryptogram and key stream as inputs. This is a mapping from $S_C \times S_K$ to S_M.

The overall block diagram of the proposed stream cipher scheme is presented in Fig. 11. Where PCA based pseudorandom pattern generator is of type shown in Fig. 9 or Fig. 10. Selector of $\lceil \log_2(sl) \rceil$-bit length (sl denotes the number of select lines to MUX (Fig. 11)) used to select any one of first, second, third and last 4-bit of the PCA. These 4-bit EXORed with plaintext (ciphertext) produce ciphertext (plaintext). The sequential steps of encryption (decryption) are noted in the Algorithm 4.

Algorithm 4:

Step 1: *Initialize and configure PCA based pseudograndom pattern generator with key.*

Step 2: *While (not end of plaintext (ciphertext)) do Step* 3 *through step* 4.

Step 3: *Generate key streams by running the generator once.*

Step 4: *EXOR* 4*-bit key stream with next* 4*-bit plaintext (ciphertext) and produce output* 4*-bit ciphertext (plaintext).*

Step 5: *Terminate (i.e., deactivate CLOCK) PCA based pseudorandom pattern generator and stop.*

VII. Invulnerability of the Scheme

Security of the schemes against possible attacks are now discussed.

A. Block Ciphers

1) Ciphertext Only Attack: As per the proposed scheme, same ciphertext may be generated from different plaintext, as well as any ciphertext may give rise to different plaintext under different CA rule configurations. Thus, the scheme is guarded against cryptanalyst's ciphertext only attack.

2) Known and Chosen Plaintext Attack: In the case of known plaintext attack, the intruder is assumed to possess a considerable length of plaintext and corresponding ciphertext. While, in case of chosen plaintext attack, the intruder is able to acquire an arbitrary number of corresponding message and cipher pairs (M, C) of his own choosing. Naturally, immunity against these attacks will directly depend on the available key space. The following theorem establishes the relationship between security and key space.

Theorem 6 [8]: A necessary condition that a cryptosystem has perfect secrecy is that it has at least as many keys as messages.

In this context, we evaluate the available key space in the proposed scheme as follows. Assume that size of the message block is N. So the number of elements in the alternating group A_{2^N} is $(2^N)!/2$. Let Z number of these elements be realized by CA with fundamental transformations and q fundamental transformation be used for one enciphering function. The number of choice of q fundamental transformations is ${}^{Z}C_q$. Again these q transformations can be arranged in $q!$ different ways. Let there be p enciphering functions stored in the ROM. So, the enciphering functions can be stored in $\prod_{i=0}^{p-1}(q! - i)$ ways. In our case, the key is nothing but the rule sequence stored in ROM. Hence, the key space is ${}^{Z}C_q \prod_{i=0}^{p-1}(q! - i)$.

Thus the key space can be made to be comparable with message size by proper choice of N, q, and ROM space.

For example, if $N = 8, Z = 184$ (as per Table III), $q = 5$ and ROM address space of 1024, then the key space of the scheme is ${}^{184}C_5 \times ((5!)!)$—an extremely large value.

B. Stream Ciphers

1) Ciphertext Only Attack: A common type of running key generators employed in stream-cipher systems consists of L (mostly maximum-length) binary LFSRs whose output sequence $(X1, X2, X3, \cdots, Xm)$ are combined by a nonlinear boolean function $\Phi[13, 14, 11]$. This is defined as

$$K = \Phi(X1, X2, X3, \cdots, Xm)$$

The inputs for the function Φ is generated by independent and identically distributed random variables. It is desirable that the combining functions should not leak information about the individual LFSR-sequences into the key stream. In this context, the concept of correlation immunity has been introduced in [13] in order to prevent divide and conquer correlations attacks. For nonlinear combiners there is a trade off between the nonlinear order of the boolean function and its

order of correlation immunity. The trade off can be avoided if the function is allowed to have memory [11].

In the proposed CA based stream cipher scheme, the key stream has been derived with the PCA based nonlinear enciphering function Φ^* as, $K_{i+1} = \Phi^*(R_i, S_i)$, where R_i is the ith rule configuration stored in ROM and S_i is the ith state of the CA. The successive states of PCA are not independent, rather they evolve from the rule R_i and S_i, which in turn depends on all previous states and rules. Also the PCA realizes different rule configurations in the same block with L memory cells. This ensures desired immunity against the schemes analogous to conventional correlation attack. However, all possible ciphertext only attacks against the proposed scheme are being evaluated.

2) Known Plaintext Attack: Cracking key stream generators amounts to the same thing as conducting known-plaintext attacks on stream ciphers. The complexity to crack key stream generators can be evaluated as follows.

a) **PCA with ROM**: The l maximal length CA can be taken from w maximal length CA in ${}^wC_l$ ways. The next state of a CA fully depends on its current state and the rule configuration. Again for maximal length CA, portion of next state does not purely depend on rule configuration. So, the complexity to get the rule sequence for a particular seed can be evaluated as,

$$\begin{aligned} &{}^wC_l \times \{l + (l-1) + (l-2) + \cdots + 3 + 2\} \\ &\quad = {}^wC_l \times \{l(l+1)/2 - 1\}. \end{aligned}$$

The PCA can be loaded with the initial nonzero seed in $2^L - 1$ ways. Hence, complexity to get the rule sequence and initial seed for the proposed key stream generator is $(2^L - 1) \times {}^wC_l \times \{l(l+1)/2 - 1\}$. For example, for $L = 8$, $w = 32$ (as per simulation) and $l = 16$, the cracking complexity is $(2^8 - 1) \times {}^{32}C_{16} \times (32 \times 31/2 - 1)$ i.e., $O(10)^{13}$. For $n = 16$, $w = 4096$ (as per simulation results) and $l = 2048$ then the cracking complexity is $(2^{16} - 1) \times {}^{4096}C_{2048} \times (2048 \times 2047/2 - 1)$.

b) **Two Stage PCA**: The number of different configurations of PCA_2 is 2^L. Initial seed of both PCA_1 and PCA_2 can be loaded in $(2^L - 1)$ ways. Hence the worst case complexity to get the key is $2^L(2^L - 1)(2^L - 1)$. For example, in case of $L = 16$ the complexity of the order of $O(10^{14})$.

VIII. Conclusion

In this work we have proved analytically that CA with EXNOR rules (i.e., 51, 153, and 195) can generate an alternating group. It is found that alternating groups form a set of fundamental transformations. Using these fundamental transformations a block cipher scheme is proposed in this paper. A scheme for stream ciphers is also proposed employing PCA. Keystream generators of the stream ciphers consist of PCA and rule selector. The complexity of the proposed schemes compare with the available schemes reported so far. The complexity can be further increased with an increase in block size for block cipher and in number of CA cells in the stream ciphers. Enciphering and deciphering process of the proposed schemes follow the same protocol. One of the main advantages of proposed schemes is the use of simple, regular, modular and cascadable structure of CA as the basic building block that ideally suits for VLSI implementation.

References

[1] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Physics,* vol. 55, no. 3, pp. 601–644, 1983.

[2] A. K. Das, A. Ganguly, A. Dasgupta, S. Bhawmik, and P. Pal Chaudhuri, "Efficient characterization of cellular automata," *IEE Proc.,* vol. 137, Pt. E, no. 1, pp. 81–87, Jan. 1990.

[3] A. K. Das, S. Saha, A. Roy Chowdhury, S. Misra, and P. Pal Chaudhuri, "Signature analyzer based on additive cellular automata," in *Proc. 20th Fault Tolerant Computing Syst.,* pp. 265–272, U.K., June 1990.

[4] W. Pries, A. Thanailakis, and H. C. Card, "Group properties of cellular automata and VLSI applications," *IEEE Trans. Comput.,* vol. C-35, no. 12, pp. 1013–1024, Dec. 1986.

[5] O. Martin, A. M. Odlyzko, and S. Wolfram, "Algebraic properties of cellular automata," *Commun. Math. Phys.* vol. 93, pp. 219, 1984.

[6] D. E. Denning, *Cryptography and Data Security.* Reading, MA: Addison-Wesley, January 1982.

[7] I. N. Herstein, *Topics In Algebra.* New Delhi, India: Vikas Publishing House Pvt. Ltd, 1976.

[8] D. Welsh, *Codes and Cryptography.* Oxford: Clarendon Press, 1988.

[9] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security.* Australia: Prentice Hall of Australia, 1989.

[10] A. Salomma, *Public-Key Cryptography.* Berlin Heidelberg: Springer-Verlag, 1990.

[11] W. Meier and O. Steffelbach, "Correlation properties of combiners with memory in stream ciphers," in *Proc. Advances in Cryptology–EUROCRYPT'90,* Springer-Verlag, 1990, pp. 204–213.

[12] J. Pieprzyk and X. M. Zhang, "Permutation generators of alternating groups," in *Proc. Advances in Cryptology-AUSCRYPT '90,* Springer-Verlag, 1990, pp. 237–244.

[13] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications," *IEEE Trans. Inform. Theory,* vol. IT-30, no. 5, pp. 776–779, Sept. 1984.

[14] ——, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Trans. Comput.,* vol. C-34, no. 1, pp. 81–85, Jan. 1985.

[15] P. H. Bardell, "Analysis of cellular automata used as pseudorandom pattern generators," in *Proc. Int. Test Conf.,* 1990, pp. 762–768.

[16] A. K. Das and P. Pal Chaudhuri, "Vector space theoretic analysis of additive cellular automata and its application pseudo-exhaustive test pattern generation," *IEEE Trans. Comput.,* vol. 42, no. 3, pp. 340–352, Mar. 1993.

[17] D. Roy Chawdhury, I. Sengupta, S. Basu, and P. Pal Chaudhuri, "Cellular automata based error correcting codes (CAECC)," *IEEE Trans. Comput.,* vol. 43, no. 6, pp. 759–764, June 1994.

[18] D. E. Knuth, *The Art of Computer Programming—Seminumerical Algorithms.* Reading, MA: Addison-Wesley, 1981.

[19] S. Even and O. Goldreich, "DES-like functions can generate the alternating group," *IEEE Trans. Inform. Theory,* vol. IT-29, no. 6, pp. 863–865, Nov. 1983.

[20] P. D. Hortensius, R. D. Mcleod, W. Pries, D. M. Miller, and H. C. Card, "Cellular automata based pseudorandom number generators for built-in self-test," *IEEE Trans. Comput.-Aided Design,* vol. 8, no. 8, pp. 842–59, Aug. 1989.

[21] Ph. Tsalides, T. A. York, and A. Thanailakis, "Pseudorandom number generators for VLSI systems based on linear cellular automata," in *IEEE Proc. E. Comput. Digit. Tech,* vol. 138, no. 4, 1991, pp. 241–249.

[22] Ph. Tsalides, "Cellular automata based built-in self test structures for VLSI systems," *Electron. Lett.,* vol. 26, no. 17, pp. 1350–1352, 1990.

[23] T. K. York, Ph. Tsalides, B. Srisuchinwong, P. J. Hicks, and A. Thanailakis, "Design and VLSI implementation of a mod-127 multiplier using cellular automaton-based data compression techniques," in *IEEE Proc. E. Comput. Digit. Tech.,* vol. 138, no. 5, 1991, pp. 351–356.

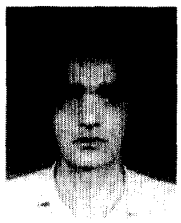
[24] P. Tzionas, Ph. Tsalides, and A. Thanailakis, "Design and VLSI implementation of a pattern classifier using pseudo @D cellular automata," *IEE Proc. G,* vol. 139, no. 6, pp. 661–668, Dec. 1992.

[25] B. Srisuchinwong, Ph. Tsalides, T. A. York, P. J. Hicks, and A. Thanailakis, "VLSI implementation of mod-p multipliers using homomorphisms and hybrid cellular automata," *IEE Proc. E,* vol. 139, no. 6, pp. 486–490, Nov. 1992.

[26] S. Wolfram, "Cryptography with cellular automata," in *Advances in Cryptology—Crypto'85* (Springer-Verlag Lecture Notes in Computer Science 218), 1986, pp. 429–432.

S. Nandi was born in West Bengal, India in 1962. He received the B.Sc. (Honours) degree in physics in 1984, B.Tech in instrumentation in 1987 and M.Tech in computer science and engineering in 1989, all are from Calcutta University, India. Currently he is a Ph.D. candidate in the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India.

During 1989–1990, he was a faculty at the Computer Engineering Department, Birla Institute of Technology, Mesra, India. Since January 1991, he has been associated with the VLSI CAD research project activities, sponsored by the Department of Electronics, Government of India. His research interests include design for testability, BIST, error correcting code, and data encryption.



B. K. Kar received the Ph.D. degree in applied mathematics from the Indian Institute of Technology, Kharagpur, India in 1989.

Subsequently, he joined the project staff in the VLSI CAD research project (sponsored by the Department of Electronics, Government of India) in the Computer Science and Engineering Department of IIT, Kharagpur. His major research interests are in the areas of cellular automata, data encryption, neural computing and fuzzy systems.

P. Pal Chaudhuri received the B.E. degree in electrical engineering from Bengal Engineering College, West Bengal, India in 1963 and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, India in 1979.

From 1963 to 1975, he was associated with IBM World Trade Corporation, India in various capacities. Subsequently, he switched over to an academic profession, starting his career as Assistant Professor at the Indian Institute of Technology, Kharagpur, India. He took the leading role in establishing the department and the undergraduate program in Computer Science and Engineering. Since 1980, he has held the position of Professor. During the period 1984–1985, he was on leave from the Institute to take up the post of Director at the Regional Computer Center, Calcutta, an autonomous scientific society. From 1986 to 1989, he was the head of the Department of Computer Science and Engineering, IIT, Kharagpur. He was closely associated with the growth of computer science and engineering education in various institutes in India. Since 1987, he has headed the VLSI CAD research activities, sponsored by the Department of Electronics, Government of India. During the period 1991–1992, he undertook two outside assignments—Visiting Professor at the University of Illinios, Urbana-Champaign, and Technical Advisor to Cadence Design Systems (India).

His major research interests include several aspects of VLSI design (mainly high level synthesis, synthesis for testablity, VLSI testing) and study of the theory and application of homogeneous structures like cellular automata in various fields—BIST, VLSI design, error-correcting codes, data encryption, data compression, etc.