Indian Journal of Science and Technology

Supplementary Article

An Application of Non-Uniform Cellular Automata for Efficient Cryptography

A. Kumaravel[1*] and Oinam Nickson Meetei[2]

[1] Dean & Professor, Department of Computer Science & Engineering
Bharath University, Chennai; drkumaravel@gmail.com
[2] PG Scholar, Department of Computer Science & Engineering
Bharath University, Chennai; nickson_s@hotmail.com

Abstract

One of the core issues for robust cryptographic systems is to deal with the rate of diffusion and distribution of keys. We try to strengthen this aspect by increasing the length of block ciphers. We present a new encryption and decryption algorithm for block cipher that supports 2^7 bit block size. All components in our system are chosen to be based on cellular automata so as to achieve higher parallelism and to simplify the in hardware and software implementation for applications with high degree of security. The main objective of this paper is to increase the complexity by novel schemes of mixing (reversible cellular automata) RCA and (non-uniform reversible cellular automata) NRCA. We apply set of different bit permutation methods for this purpose .This paper establishes the proof for existence of yet another approach for a high quality pseudo-random bit sequences generated by non uniform cellular automata.

Keywords: Cryptographic System, NRCA-non Reversible CA, Pseudorandom Number, RCA-Reversible Cellular Automata.

1. Introduction

Two main cryptography systems are used today: symmetric systems (aka secret key systems) and public-key systems. An extensive overview of currently known or emerging cryptography techniques used in both type of systems can be found in [1, 9]. One of such a promising cryptography technique is applying cellular automata (CAs).

The main concern of this paper is secret key systems. In such systems the encryption key and the decryption key are same (symmetric key). The encryption process is based on generation of pseudorandom bit sequences, and CAs can be effectively used for this purpose. Cellular Automata (CA) is an organized lattice of cells and each cell have finite number of states, such as "TRUE" (T) or "FALSE" (F). The lattice dimensions can be of any finite value. Each cell within a collection of cells is called as hood. It is characterized relatively with respect to a particular cell. To start with at time t=0, a state is assigned to the cells. The new states of the cell depend on its own previous state and states of its neighborhood. The new states are assigned based on some predefined rule using mathematical calculations.

Encryption, by theory requires highly complex actions such as permuting, flipping and altering data in such a way that it is undecipherable and provides complex relationship with the original text and the keys. This relationship should be non-linear so that decryption is as tough as possible. The encryption process must be faster in time and cheaper in terms of the components involved [2, 7]. CA provides a basic structure for highly parallel and complex operations upon which a basic encryption scheme can be built. The message encryption is done by Pseudo Random Number Generators (PRNGs) using CA. The generation of new states in One-Dimensional (1-D) CA, can be considered as a sequence of

* *Corresponding author:*
A. Kumaravel (drkumaravel@gmail.com)

random numbers [3]. Different security schemes have been proposed including symmetric key, hash functions and public key cryptography as observed by Sarkar, [3]. Further as stated by Wolfram [4, 5], Rule 30 promotes the use of large integers in the pseudo random number generation. Owing to this interesting chaotic property of the peculiar CA, Wolfram states that, this kind of CA is used as random number generator.

The structure of the paper is as follows: Section II describes studies of symmetric key algorithms and its performances; Section III depicts our proposed algorithm in detail. Finally, conclusion is expressed.

2. Terms, Models and Methods

Here we discuss the terminology used in the paper for easier reading.

2.1 Lightweight Cellular Automata Symmetric-key Encryption

One-dimensional cellular automata encryption algorithm called as Lightweight Cellular Automata-based Symmetric-key Encryption (LCASE) is described in [2]. Here rule 30 is used for encrypting plain-text with keys. The fundamental design objective of LCASE is to effectively optimize the performance requirements of both software and hardware. However, the algorithm also has to address the conventional security constraints. Different issues considered for designing the proposed algorithm are:

a. High speed performance and minimum code density
b. Defiant to attack such as conventional cryptanalysis and timing attack
c. Simple and code effective implementation

The implementation of cipher in either of hardware or software has to satisfy two important design considerations [2]. The algorithm should be fast and easy to implement. In order to address these constraints parallel processing methods are used. CA elements are found to be effective in the design process. Number of loops used in LCASE consists of two rounds namely, 1. Last-half (LH) 2. First- half (FH). The term 'r' refers to a complete round and 'r+1' round refers to FH. The selection of 'r' value is shown in the Table 1.

This encryption is a reversible two way process, decryption is obtain as the result of this. Each encryption round uses 1-D (3-neighbourhood) 32 bits periodic boundary CA. The basic element of a round uses Reversible Cellular Automata (RCA), Bit Permutation (BP), Non-autonomous Cellular Automata (NCA) and Reverse Substitution (RS). The rules used in LCASE for different operations are given in Table 2.

Table 1. Selection of 'r' value

Keys size in bits	128	192	156
'r' value	12	14	16

Table 2. Selection of rule for different operations

Operation	Rule
RCA(second order)	30
	Skewed 30
	Skewed 45
NCA	218

In order to effectively obtain the non linearity property, suitable rules from Table 2 have to be chosen for the selected operators. In this cipher, RCA and NCA are the preferred operators over the conventional XOR, as the later is not suitable for non linear operations.

3. Main Algorithm

We construct the main algorithm based on following terminologies.

3.1 Next State Generation by Various Ca Rules

Our proposed method uses CA rules 30, 45,218 as mentioned as in [2] and 86, 90,105,150,165 as in [6] for NRCA and RCA operation. The next state generation of these 8 rules is given in Table 3.

Table 3. Next State Configuration for CA Rules 30, 45, 218, 90,150,145,165 and 86

Neigh-borhood State:	111	110	101	100	011	010	001	000	Rule No.
Next State :	0	0	0	1	1	1	1	0	30
Next State :	0	0	1	0	1	1	0	1	45
Next State :	0	1	0	1	0	1	1	0	86
Next State :	0	1	0	1	1	0	1	0	90
Next State :	0	1	1	0	1	0	0	1	105
Next State :	1	0	0	1	0	1	1	0	150
Next State :	1	0	1	0	0	1	0	1	165
Next State :	1	1	0	1	1	0	1	0	218

3.2 RCA Key Schedule

RCA is the one in which the preceding pattern can be recovered from the given current pattern (s). Our proposed algorithm utilizes RCA (second order) [10]. The ‘ith’ state of n bit pattern is determined by the clock cycles as shown in Table 4. Given below is an example of a 3-neighborhood second order RCA [2]

$$x_i\,(t - 1) = f\,(x_{i-1}\,(t),(\,x_i\,(t),\,x_{i+1}\,(t))\,)\mathrm{XOR}\;x_i\,(t + 1)$$

$$x_i\,(t + 1) = f\,(x_{i-1}\,(t),(\,x_i\,(t),\,x_{i+1}\,(t))\,)\mathrm{XOR}\;x_i\,(t - 1)$$

Here, the states $x_i(t + 1)$ and $x_i(t - 1)$ are denoted respectively by the terms ξ_i and y_i and ‘f’ is from Table 5.

ξ is obtained based on two initial patterns of (Y, X) at time steps (t - 1) and t. Then, using two successive patterns (ξ, X), the initial pattern Y can be figured out. This operation is denoted as follows [2].

$$\xi = \mathrm{RCA}\,(Y, X);\; Y = \mathrm{RCA}\,(\xi, X).$$

Elementary CA rule 30 based second order periodic boundary 4-cell RCA logic diagram is depicted in Figure 1.

The RCA key schedule generates 16 bytes of key for each encryption and decryption round. For each round of key generation we applied different rule which is selected sequentially from Table 5. RCA key schedule is explained in “Figure 2” and the RCA operation between two 8 bits sub- block is shown in Figure 3.

Table 4. State Determination

Clock cycles	‘ith’ state of n bit pattern
t + 1	States of neighbourhood pattern at ‘t‘ & current cell at (t - 1)
t – 1	patterns at t & (t + 1) clock cycle

Table 5. Next State Function for cellular Automata Rules from Table 3

Rule No.	Next state function(f)
30	$x_i(t{+}1) = x_i{-}1(t)\ \mathrm{XOR}\ (x_i(t) \vee x_{i+1}(t))$
45	$x_i(t + 1) = x_{i-1}(t)\ \mathrm{XOR}\ (x_i(t) \vee x\neg_{i+1}(t))$
86	$x_i(t + 1) = x_i\neg(t) \wedge (\,x_{i-1}(t)\mathrm{XOR}\ x_{i+1}(t)) \vee x_i(t) \wedge x\neg_{i+1}(t)$
90	$x_i(t + 1) = x_{i-1}(t)\ \mathrm{XOR}\ x_{i+1}(t)$
105	$x_i(t + 1) = x_i(t)\ \mathrm{XNOR}\ (x_{i-1}(t)\ \mathrm{XOR}\ x_{i+1}(t))$
150	$x_i(t{+}1) = x_{i-1}(t)\ \mathrm{XOR}\ x_i(t)\ \mathrm{XOR}\ x_{i+1}(t)$
165	$x_i(t + 1) = x_{i-1}(t)\ \mathrm{XNOR}\ x_{i+1}(t)$
218	$x_i(t + 1) = (x_i(t) \wedge (x_{i+1}(t)) \vee (x_{i-1}(t)\ \mathrm{XOR}\ x_{i+1}(t))$

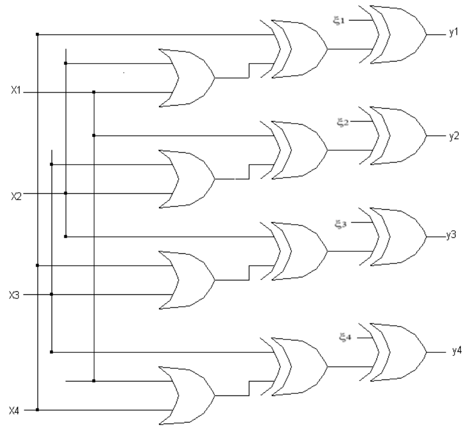
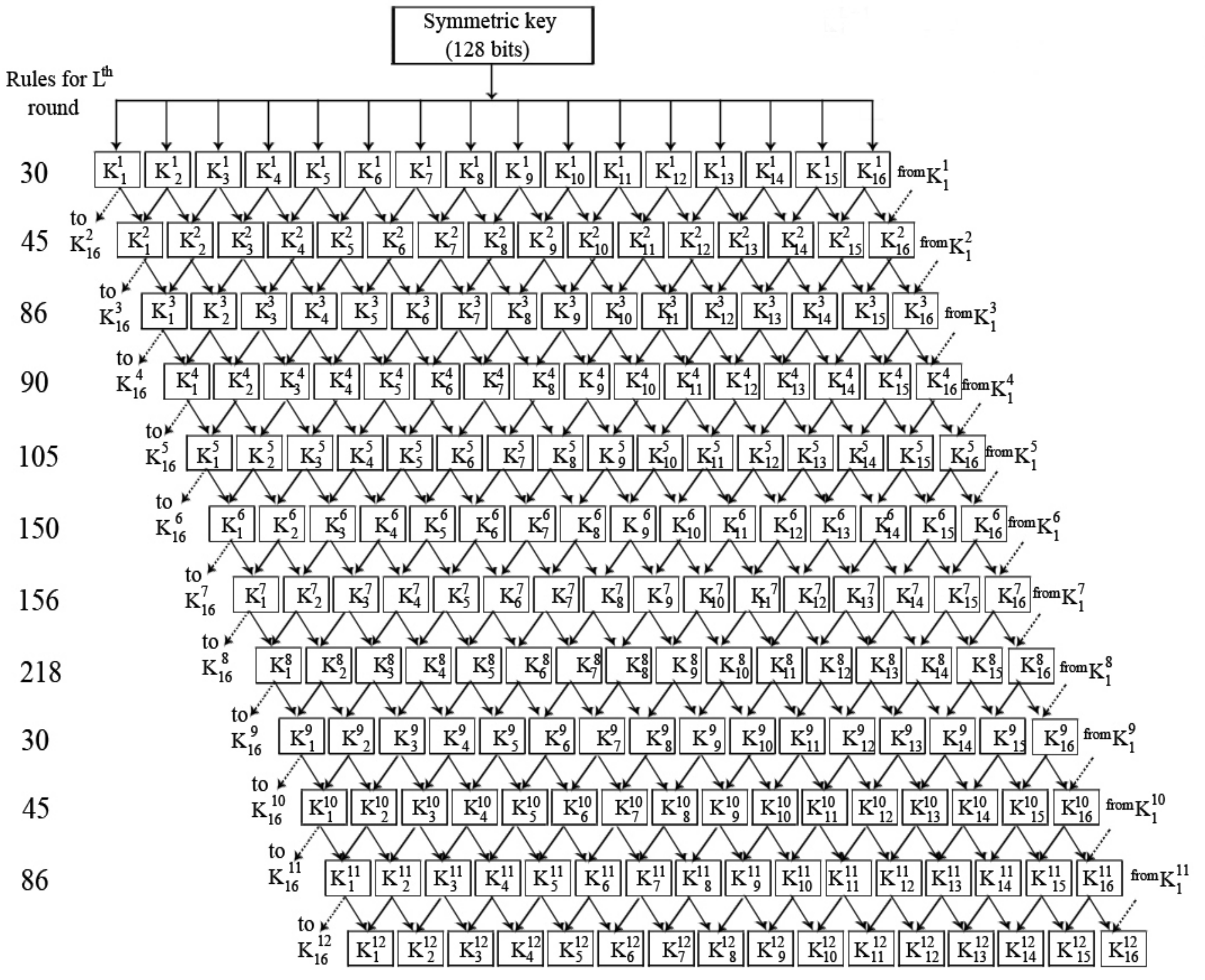


Figure 1. Combintiraal circuit for 4-cell periodic boundary RCA based on. Rule 30, represented by the state equation $x_i\,(t + 1) = f(x_{i-1}\,(t)\mathrm{XOR}(\,x_i\,(t) \vee x_{i+1}\,(t))\,)\mathrm{XOR}\;x_i\,(t - 1)$

Pseudocode:

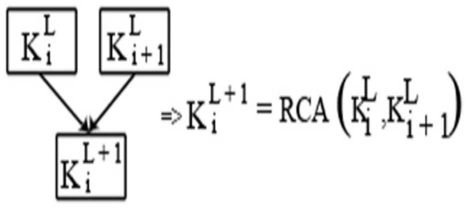
Pseudocode for RCA key schedule in Figure 2.

```
Input: K($k_i$ : i:=1 to 16);//K=user selection 128 bits key and $k_1,k_2,\ldots\ldots,k_{16}$ are 8 bits words//
Output: $k_i^L$ (i=1 to 16);// $k_1^L,k_2^L,\ldots\ldots,k_{16}^L$ are 8 bits words key schedule generated for L^th round//
L= number of rounds
          BEGIN
            // computation of encryption key//
               for r:=1 to L-1 do
                  for i:=1 to 16 do
                     if (i:=16) do
                        $K_i^{r+1}$:=RCA($K_i^r$ ,$K_1^r$);
                     endif
                     if(i >16) do
                        $K_i^{r+1}$:=RCA($K_i^r$ ,$K_{1+1}^r$);
                     endif
                  endfor
               endfor
//Determination of decryption key //
     //DK= decryption key and k= encryption key//
               For r:=1 to 16 do
                  $DK_1^r$:=$k_1^{|r-(L+1)|}$;
               endfor
            END
//For each round of encryption and decryption the RCA key schedule generates sixteen keys each of eight bits.
The next key schedule is generated from the previous key schedule.//
```



K_i^L= ith 8 bits sub block for Lth round.

Figure 2. RCA key schedule.



K_i^L= ith 8 bits sub block for Lth round.

Figure 3. RCA operation between two 8 bits sub- block.

3.3 Modulo Prime Operation for Permutation

The Bit-Permutation (BP (input[32 bits])) operation permutes the ith bit into ((9* i mod 31)+1)th bit. The idea behind this permutation is to place the three consecutive bits into three different locations. This increases the rate of diffusion and makes differential cryptanalysis very complex for our encryption scheme. Moreover the realization of permutation is obtained by connecting wire crossings appropriately.

Figure 4 shows an example of bit permutation which involves 4 bytes (32 bits). Here the three neighborhoods bits in position 1,2 and 3 are transferred into 10, 19 and 28 positions.

3.4 Non-Uniform Reversible Cellular Automata

NRCA is a RCA operation in which we applied 8 different rules to each bit of each byte from Table 5. Each bit being applied with a unique rule. The transition function are computed parallel in all bits as shown in the Figure 5 Each block receives 8 bits and produces 8 bits.

3.5 Reversible Bit Permutation (RBP)

The Reversible Bit-Permutation (RBP) operation is the process of re-permuting the distributed bits into its original position. The idea behind is to restore original position.

Figure 4 shows an example of bit permutation which involves 4 bytes, in RBP, the three distributed bit in position10, 19 and 28 is transfer into its original position that is 1, 2 and 3 position as shown in Figure 6.

Algorithm for Reversible bit permutation.

Step 1: Assign j from 1 to 31 in the input word RBP input [32 bits]

Step 2: Compute k as ((9*j mod 31)+1)

Step 3: RBP output[j] = RBP input[k]

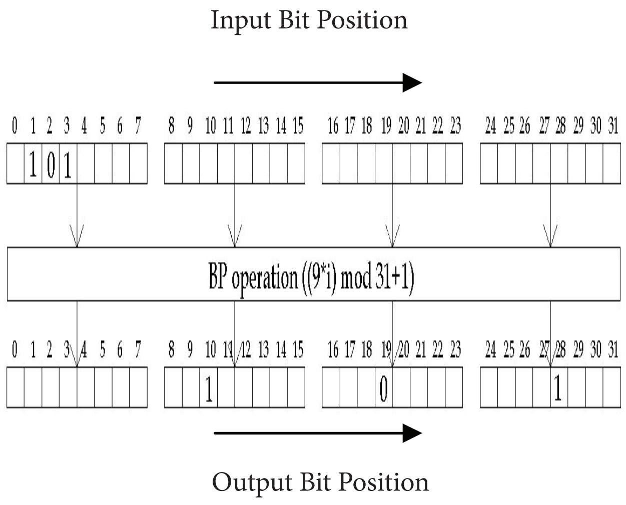
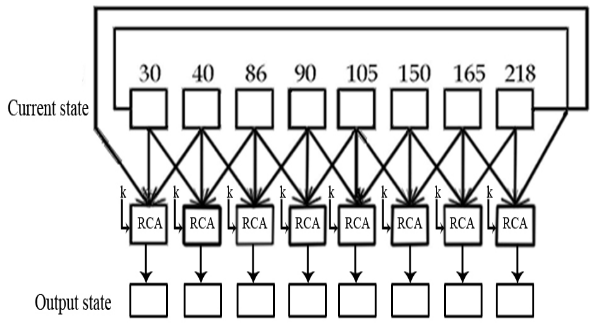


Figure 4. Bit permutation in 32 bits.



30,40,86,90,105,150,165 are rules use in RCA
K = encryption key for the round for the block

Figure 5. NRCA operation in 8 bits block.

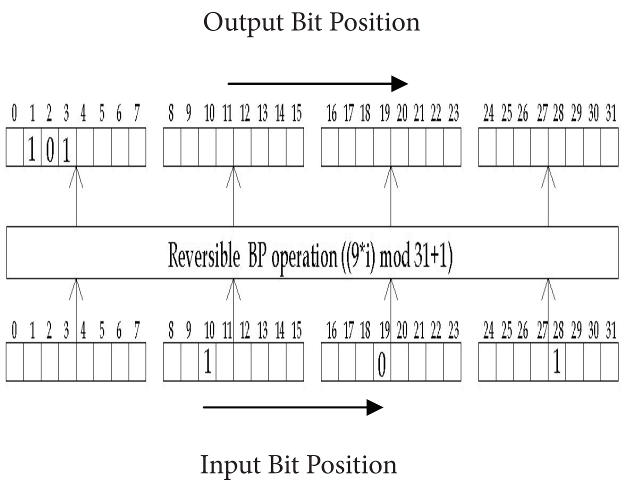


Figure 6. Reversible bit permutation.

3.6 Main algorithm

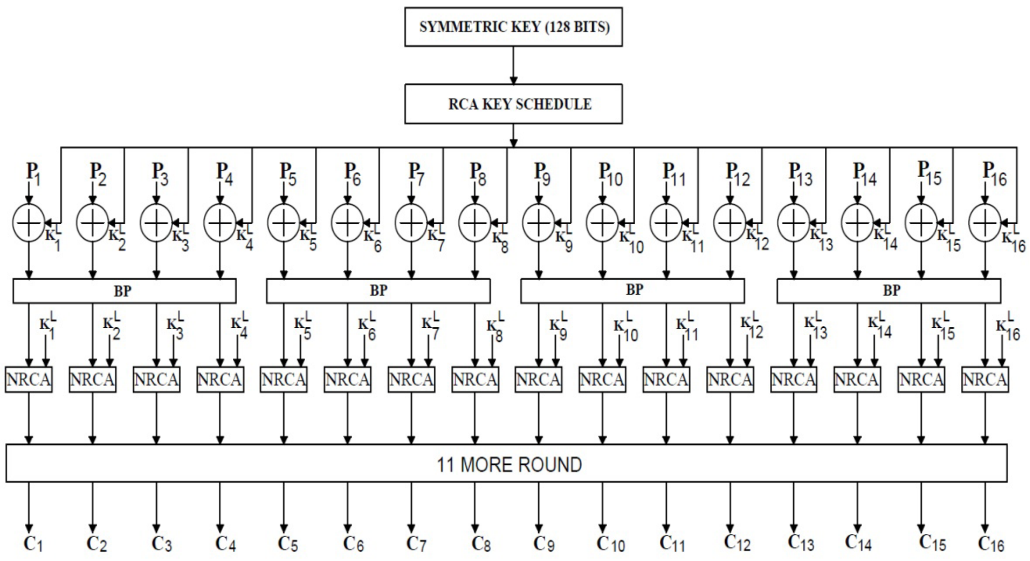
Using the above method we describe the encryption process with one-dimensional reversible cellular automata (1D RCA) as shown in Figure 7. Our entire encryption round comprises simple XOR, Bit Permutation, RCA key schedule, NRCA. The 'r' values are chosen from the Table 1. Here we used 8 different rules from Table 3 for NRCA and RCA operation . Each encryption round uses the 8 different rules to each byte for NRCA operation. In RCA key generator, for each encryption round a rule is selected sequentially in cyclic order from the Table 3 for generating next key.

3.7 Decryption

Since we used RCA (reversible cellular automata), NRCA(non-uniform RCA), XOR and reversible BP(bit permutation), we can decrypt the cipher text to plain text in reverse way using the decryption key which is explained in pseudo code of RCA key schedule. The cipher text can be decrypted from the cipher text and the key. The computational flow of decryption scheme is given in Figure 8.

4. Conclusion

The above routines are tested and we have comparable results based on these proposed algorithms. In this paper we have tried with a light weight cryptographic algorithm based on symmetric key based on non-uniform cellular automata and dynamic key generation. This approach decreases the space requirements especially for small devices and increases the better results with the rate of


SYMMETRIC KEY (128 BITS)
RCA KEY SCHEDULE
P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16}
K^L_1 K^L_2 K^L_3 K^L_4 K^L_5 K^L_6 K^L_7 K^L_8 K^L_9 K^L_{10} K^L_{11} K^L_{12} K^L_{13} K^L_{14} K^L_{15} K^L_{16}
BP
NRCA
11 MORE ROUND
C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14} C_{15} C_{16}

P_i: ith 8-bit plain text sub-block
K_i^L: ith 8-bit key sub-block in Lth round
BP: Bit Permutation
NRCA: Non-uniform RCA
c_i :ith 8-bit cipher sub-block
$\oplus$: Boolean XOR operation
RCA: Reversible CA

Figure 7. 128 key bits Encryption scheme applying CA rules.

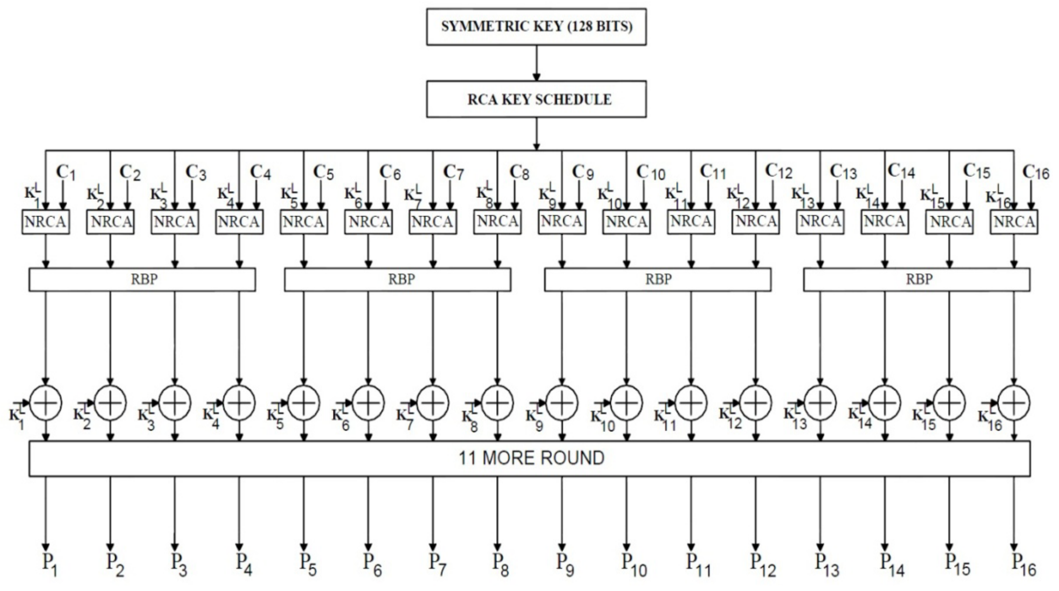
Pseudocode:
Pseudo code for realising the CA rules found in Figure 6.

```
// Uppercase alphabet set represent 128 bit block and corresponding lowercase sets specify 8 bit words.//
    Input:Plaintext block, P($p_i$: i=1 to 16);
                K: User 128 bits key;
Output: Cipher block, C($c_i$: i= 1 to 16);
Intermediate cipher text : IC($Ic_i$: i=1 to 16);
$BP_1$, $BP_2$, $BP_3$, $BP_4$ are 32 bits words
$bp_1$,$bp_2$.........$bp_{16}$ 8 bits words

    BEGIN
        READ $p_i$;
        READ K;
        //Apply RCA key schedule//
        IC:=P;
        Keyschedule($k_i$) , (i=1 to 16); //16 8 bits block generated by key generator for Lth round//
        For L:=1 to end of round do
        $S_1$:=$Ic_1 \oplus k_1$; $S_2$:=$Ic_2 \oplus k_2$; $S_3$:=$Ic_3 \oplus k_3$;
        $S_4$:=$Ic_4 \oplus k_4$; $S_5$:=$Ic_5 \oplus k_5$; $S_6$:=$Ic_6 \oplus k_6$;
        $S_7$:=$Ic_7 \oplus k_7$; $S_8$:=$Ic_8 \oplus k_8$; $S_9$:=$Ic_9 \oplus k_9$;
        $S_{10}$:=$Ic_{10} \oplus k_{10}$; $S_{11}$:=$Ic_{11} \oplus k_{11}$; $S_{12}$:=$Ic_{12} \oplus k_{12}$;
        $S_{13}$:=$Ic_{13} \oplus k_{13}$; $S_{14}$:=$Ic_{14} \oplus k_{14}$; $S_{15}$:=$Ic_{15} \oplus k_{15}$;
        $S_{16}$:=$Ic_{16} \oplus k_{16}$;

    // create four $BP_i$//

        $BP_1$:=BP($S_1$,$S_2$,$S_3$,$S_4$); //$S_{1,......}$,$S_{16}$ are 8 bits wards//
        $BP_2$:=BP($S_5$ ,$S_6$,$S_7$,$S_8$);
        $BP_3$:=BP($S_9$,$S_{10}$,$S_{11}$,$S_{12}$);
        $BP_4$:=BP($S_{13}$ ,$S_{14}$,$S_{15}$,$S_{16}$);
    //corresponding lowercase specify 8 bit//
        $BP_1$($bp_1$,$bp_2$,$bp_3$,$bp_4$);
        $BP_2$($bp_5$,$bp_6$,$bp_7$,$bp_8$);
        $BP_3$($bp_9$,$bp_{10}$,$bp_{11}$,$bp_{12}$);
        $BP_4$($bp_{13}$,$bp_{14}$,$bp_{15}$,$bp_{16}$);
    //corresponding intermediate cipher text//
        for i: = 1 to 16 do loop
            $Ic_i$:= NRCA($bp_i$,$k_i$);
          end loop
        endfor
        for i:=1 to 16 do
          $c_i$:=$Ic_i$;
        endfor
        END
```



P_i: ith 8-bit plain text sub-block
K_i^L: ith 8-bit key sub-block in Lth round
RBP: Reversible Bit Permutation
NRCA: Non-uniform RCA

c_i : ith 8-bit cipher sub-block
⊕ : Boolean XOR operation
RCA: Reversible CA

Figure 8. 128 key bits Decryption scheme applying CA rules.

diffusion and distribution of keys. Moreover the different rule of neighborhood has been applied for cellular at each byte position for encryption and decryption. Hence our proposed method strengthens the simplicity for establishing the implementation in software and hardware.

5. References

1. Schneier B (1996). Applied cryptography, Wiley, New York.
2. Tripathy S, and Nandi S (2009). LCASE: Lightweight Cellular Lightweight Cellular Automata-based Symmetric-key Encryption, International Journal of Network Security, vol 8(2), 243–252.
3. Sarkar P (2000). A brief history of cellular automata, Journal of ACM Computing Surveys (CSUR), vol 32(1), 80–107.
4. Wolfram S (1986). Cryptography with cellular automata, Crypto '85, LNCS 218, Springer-verlag, 429–432.
5. Wolfram S (1986). Random sequence generation by cellular automata, Advances in Applied Mathematics, vol 7(2), 123–169.
6. Seredynski F, Bouvry P et al. (2004). Cellular automata computations and secret key cryptography, Parallel Computing Journal, vol 30(5-6), 753–766.
7. Standaert F, Piret G et al. (2004). ICEBERG : An involutional cipher efficient for block encryption in reconfigurable hard- ware, FSE '04, LNCS 3017, Springer- verlag, 279–299.
8. Sklavos N, Moldovyan N A et al. (2005). High speed networking security: Design and implementation of two new DDP-based ciphers, Mobile Networks and Applications-MONET, vol 25(1-2), 219–231.
9. Moldovyan N A, Moldovyan P A et al. (2007). On software implementation of fast DDP-based ciphers, International Journal of Network Security, vol 4(1), 81–89.
10. Toffoli T, and Margolus N (2001). Invertible cellular automata: A review, Physica D, vol 45(1–3), 229–253, (Reprinted with correction as of Oct).