LCASE: Lightweight Cellular Automata-based Symmetric-key Encryption

Somanath Tripathy[1] and Sukumar Nandi[2]
(Corresponding author: Somanath Tripathy)
Indian Institute of Information Technology and Management, Gwalior (Email: somanath.tripathy@gmail.com)[1]
Indian Institute of Technology, Guwahati (Email: sukumar@iitg.ac.in)[2]
(Received Feb. 21, 2008; revised May 9, 2008; and accepted May 30, 2008)

Abstract

We propose a lightweight block cipher that supports 128-bit block size with 128-, 192- and 256-bit keys, to confirm with the Advanced Encryption Standard (AES) specification. All components of LCASE are chosen to be cellular automata-based so as to achieve higher parallelism and simplify the implementation. Apart from that, the other virtues of LCASE are its high speed and cheap cost along with being resistant against timing analysis attacks.

Keywords: Block cipher, cellular automata, lightweight cryptography

1 Introduction

Recent advancement of communication and computing technologies introduces different types of portable devices that populate in our day to day life. These devices have limited battery power, restricted storage and low computation power to bring the device in affordable cost and portable size. Information security is of primary concern for all users irrespective of the computing device being used. Among the different approaches for achieving information security, the present work concerns with symmetric key cryptography [15]. Variety of encryption algorithms are available to encrypt the data. Execution of the traditional encryption algorithms consume time, space and energy. Moreover, side channel attacks are based on time and power that can be applied to the block ciphers implemented on smart card technology [12]. Also protecting implementation against these kinds of attacks is usually difficult and expensive.

Application of cryptographically strong algorithm such as AES-Rijndael [8] leads to significant transmission delay, and require high computations as well as large storage capacity. It leads to infeasible to incorporate the strong cryptographic algorithms in the resource constrained devices. On the other hand, encryption algorithm ICEBERG [24] proposed for its implementation with special emphasis to the reconfiguration hardware. However, the software implementation is not suitable i.e., not cost effective with respect to storage requirement and/ or speed.

Data dependent permutation (DDP)-based fast encryption algorithms are appeared to be faster and efficient for high speed networks. Recently, Cobra-H64 and Cobra-H128 were proposed in [22] use switchable operations to prevent the weak keys identified against the earlier DDP-based encryption techniques. These ciphers are specially emphasized for high speed performance hardware implementation but requires more hardware resources. On the other hand, Cobra-S128 [16] proposed for software implementation uses addition and subtraction modulo 2^{32} operations. These DDP -based techniques even though are optimized for hardware or software implementation, consumes more resources (area,storage, computation power) lead them unsuitable for implementing in resource constrained devices. A suitable alternative to this is to use techniques those are fast in terms of parallel operations and lightweight in terms of both, computation and storage requirements.

The inherent parallelism property of Cellular Automata (CA) has been exploited by several researchers to design high speed encryption schemes [18, 27]. The cipher proposed in [27] can not guarantee security unless the size of key is so high. Because the same rule is applied repeatedly, the seed value *i.e*, the secret key can be determined easily [11]. The cipher proposed in [17] was broken in [5] due to the affine property of used CA in [19]. Recently, a reversible CA (RCA) based encryption algorithm is proposed in [19] that satisfies the strict avalanche criteria, but trades off with additional communication overhead. In [18] a CA based cryptosystem (CAC) is proposed, where non-linearity is achieved by intermixing affine CA with non-affine transformations. CAC uses two CAs called major CA and minor CA, the later one used to transform a secret key into a secret state by which the major CA is operated. Unfortunately, the CAC scheme lacks in detail on how to construct the minor CA, major CA and has been broken successfully in [2] using a few chosen plaintexts. It is due to the fact that the whole encryption operation can be converted to an

equivalent transformation in which the secret key components linearly involved with. Thus, none of these works is reported to propose a secure and resource efficient crypto-algorithm. The main reason for non-satisfying functionalities of the above mentioned cryptosystems is basically due to the structure of encryption technique which cannot be attributed to the characteristics of CA. Instead, CA can be used as a high speed cipher constructing device exploiting its complex and random output generation capability.

In this paper, we propose a resource efficient, easily realizable, and faster symmetric-key cryptosystem called LCASE (Lightweight Cellular Automata-based Symmetric-key Encryption). A rudimentary idea of this work, without taking into account the proper key scheduling and security analysis, has been presented in [26]. The proposed block cipher is simpler to implement in both hardware and software, along with being resistant against timing analysis attack. Furthermore, the resulting design does not require any cost-effective S-box which needs extra storage. Nevertheless, implementation of (non-optimized code) LCASE is faster than (optimized code) AES-Rijndael [7].

The paper is organized as follows. The Section 2 discusses briefly about CA. Section 3 discusses the design rationale which is chosen for the proposed cryptosystem. Operational details of the proposed encryption process and key schedule are presented in Section 4. Security of the proposed technique, against various cryptanalysis attacks is discussed in Section 5. Section 6 shows the effectiveness of the implementation aspect of LCASE by comparing with some of the existing schemes.

2 Cellular Automata (CA)

Cellular automata (CA) is a dynamic system which consists of a number of identical cells. The states of each cell are updated synchronously at discrete time steps according to a local update rule. This local rule is a function of the present state of its neighbors. For instance, in a 2-state 3-neighborhood CA, the evolution of i^{th} cell (x_i) of X can be formulated as a function of the present state of $(i-1)^{th}, i^{th}$, and $(i+1)^{th}$ cells; $x_i(t+1) = f(x_{i-1}(t), x_i(t), x_{i+1}(t))$. The vector $X^{(t)} = ({x_1}^{(t)}, {x_2}^{(t)}, \ldots, {x_n}^{(t)})$ is called the configuration at time t of the n-cell CA. Here, f denotes the next state function. There are $2^{2^3} = 256$ possible different next state functions for a 2-state, 3-neighborhood CA. Each of these next state function specified by a decimal number, is called as rule number. Table 1 shows the next states computed according to rule 30, rule 45 and rule 218. The topmost row shows all possible 8 configurations at instant t. The states at the instant of time $(t+1)$ are computed according to the rules as given in subsequent rows of the table.

Denoting the symbols $\neg, \vee, \wedge$, and $\oplus$ respectively for logical NOT, OR, AND and XOR operations, the Equations (1), (2), and (3) are respectively representing the combinational logics for rule 30, rule 45 and rule 218 CA. It is evident from Table 1.

$$x_i(t+1) = x_{i-1}(t) \oplus (x_i(t) \vee x_{i+1}(t)); \quad (1)$$
$$x_i(t+1) = x_{i-1}(t) \oplus (x_i(t) \vee \neg x_{i+1}(t)); \quad (2)$$
$$x_i(t+1) = (x_i(t) \wedge x_{i+1}(t)) \vee (x_{i-1}(t) \oplus x_{i+1}(t)). \quad (3)$$

Preserving the inheritance property of CA the 3-neighborhood dependency positions can be altered. The i^{th} bit of the resultant configuration depends on $i^{th}, (i+1)^{th}, (i+2)^{th}$ bits rather than $(i-1)^{th}, (i)^{th}, (i+1)^{th}$ bits in customized CA. This form of CA with skewed_rule 30 and skewed_rule 45 can be defined as

$$x_i(t+1) = (x_i(t) \oplus (x_{i+1}(t) \vee x_{i+2}(t)); \quad (4)$$
$$x_i(t+1) = (x_i(t) \oplus (x_{i+1}(t) \vee \neg x_{i+2}(t)). \quad (5)$$

The important feature of these skewed_rules is that these operations are reversible i.e.,

$$x_i(t) = (x_i(t+1) \oplus (x_{i+1}(t+1) \vee x_{i+2}(t+1));$$
$$x_i(t) = (x_i(t+1) \oplus (x_{i+1}(t+1) \vee \neg x_{i+2}(t+1)).$$

The CA rules that involve only non-linear logic operations are called non-linear CA. On the contrary, the CA involves only XOR logic (linear operations) is called linear CA.

Multi-dimensional CA having different number of neighborhood dependencies also exist. As an example a two-dimensional CA in which the cells are arranged in a grid fashion. The next state function depends on 9-neighbors (including self). However, increasing of the number of neighborhood dependencies raises the implementation cost. Therefore, to simplify the implementation we have chosen one-dimensional 3-neighborhood CA for LCASE designing. In practice, we consider the finite n-cell CA, so the boundary conditions need to be considered. This work considers periodic boundary conditions in which, if $i \equiv j \ (mod\ n)$, then ${x_i}^{(t)} = {x_j}^{(t)}$ (more details about CA please refer [29] and [6]).

2.1 Non-autonomous CA (NCA)

Unlike the formal CA, external inputs are incorporated into each cell of NCA. Therefore, it evolves different successor configurations from a single input configuration controlled by the external input. The evolving state of i^{th} cell at time $(t+1)$, $x_i(t+1)$ from an NCA by giving external input τ_i can be expressed as Equation (6).

$$x_i(t+1) = f(x_{i-1}(t), x_i(t), x_{i+1}(t)) \copyright \tau_i$$
$$= \mathcal{F}(x_{i-1}(t), x_i(t), x_{i+1}(t), \tau_i), \quad (6)$$

where © denotes for a logical operation. The resulting state of NCA rule 218 with © as $\wedge$ (logical AND), can be expressed as

$$x_i(t+1) = ((x_i(t) \wedge x_{i+1}(t)) \vee (x_{i-1}(t) \oplus x_{i+1}(t))) \wedge \tau_i.$$

Let X and Y respectively represent the initial configuration $X(t)$ and resulting configuration $X(t+1)$. The

Table 1: Next state configuration for CA rules 30, 45 and 218

Neighborhood State:	111	110	101	100	011	010	001	000	Rule#
Next State:	0	0	0	1	1	1	1	0	30
Next State:	0	0	1	0	1	1	0	1	45
Next State:	1	1	0	1	1	0	1	0	218

operation $Y = NCA(\tau, X)$ can be expressed as Equation (7), and its logic diagram (for 4-cell periodic boundary) is depicted in Figure 1.

$$y_i = ((x_i \wedge x_{i+1}) \vee (x_{i-1} \oplus x_{i+1})) \wedge \tau_i. \quad (7)$$

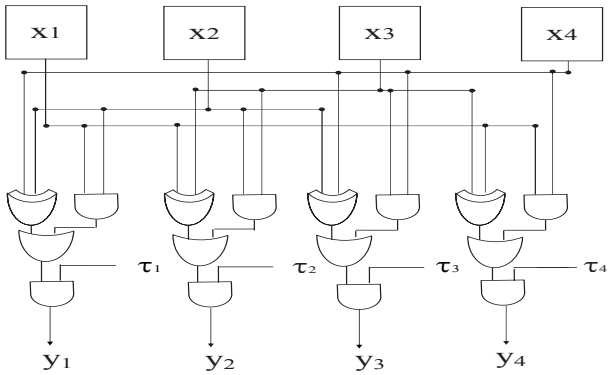

Figure 1: Logic structure of 4-cell periodic boundary NCA

Logic diagram of a second order periodic boundary 4-cell RCA using elementary CA rule 30 is depicted in Figure 2. The evolved configuration of such an RCA can be expressed as

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

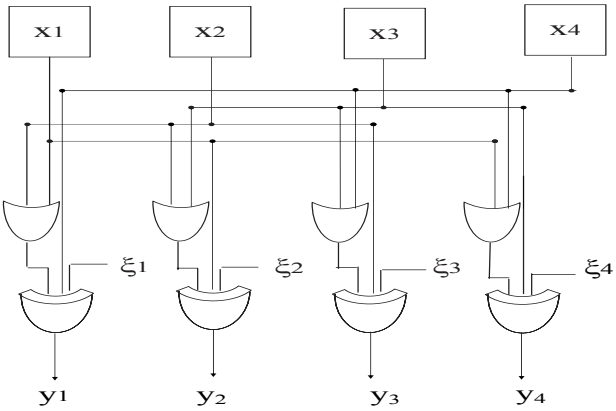

Figure 2: Logic structure of 4-cell periodic boundary RCA.

2.2 Reversible CA (RCA)

A cellular automata is said to be reversible, if the previous configurations can be retrieved from the given current configuration(s). Our proposed algorithm uses second order reversible class CA (RCA) [25] in which, the configuration of the i^{th} state on clock cycle $(t+1)$ is determined by the states of the n-neighborhood configuration at clock cycle t and the self configuration at $(t-1)$ clock cycle. In reverse, one can determine the configuration at $(t-1)$ clock cycle from the configurations at t and $(t+1)$ clock cycle. For example a 3-neighborhood second order RCA can be expressed as

$$\begin{aligned} x_i(t+1) &= f(x_{i-1}(t), x_i(t), x_{i+1}(t)) \oplus x_i(t-1); \\ x_i(t-1) &= f(x_{i-1}(t), x_i(t), x_{i+1}(t)) \oplus x_i(t+1). \end{aligned}$$

Let, ξ_i and y_i respectively denote for $x_i(t+1)$ and $x_i(t-1)$. Depending on two initial configurations (Y, X) at time steps $(t-1)$ and t, the next configuration (ξ) is evolved. Again, using two consecutive configurations (ξ, X) the initial configuration Y can be deduced. We denote these operations as

$$\begin{aligned} \xi &= RCA(Y, X); \\ Y &= RCA(\xi, X). \end{aligned}$$

3 Design Rationale

The basic design goal of LCASE is to provide an efficient algorithm for both hardware and software implementations, as well as to meet the traditional security requirements. Following criteria are taken into consideration for designing the proposed algorithm:

- Resistant against conventional cryptanalysis attacks
- Simpler to implement in hardware with reduced cost
- Easily fit into resource constraint devices
- Fast and code compactness on a wide range of platforms
- Resistant against timing analysis attack.

The cipher has been designed primarily for fast and easy implementation in both, hardware and software platforms. Parallel implementation at the bit-level can make the implementation faster. Therefore, the design choices have been related to use CA-based components.

LCASE uses second order RCA based on CA rule 30, CA skewed_rule 30, CA skewed_rule 45 and NCA based on CA rule 218. The choice of these rules has been opted from the pseudo-exhaustive simulation. It is observed that the combination of these rules provide effective non-linearity.

Key mixing in the Encryption process is not a simple XOR of the key-bits but performs through an RCA or NCA operation. RCA and NCA are the function of 3 key-bits and one text bit. This adds extra difficulty for an attacker to deduce the key from the given cipher.

4 The Proposed Scheme (LCASE)

A complete LCASE round comprises of first-half (FH) round and last-half (LH) round. The encryption algorithm is consisting of r complete rounds while the last (r+1) round is an half-round (FH-round) to make the cipher and its inverse similar structure. The values of r is chosen to be 12 / 14/ 16 respectively for 128/ 192/ 256- bit user selected keys. Each round of encryption/ decryption process uses a key sub-block that is generated from user-selected key by a simple key schedule. Detail description of the proposed encryption algorithm and key schedule are discussed subsequently.

4.1 Encryption/decryption Process

Encryption is a one-to-one function for possible decryption. Each encryption round uses one-dimensional (3-neighborhood) 32-cell periodic boundary CA. Following basic operations are involved in the encryption round.

RCA: Initial configuration of RCA is set up with keyword ${k_j}^l$, where ${k_j}^l$ denotes the j^{th} key sub-block for l^{th} round $(1 \leq l \leq r+1)$. The previous configuration of that RCA is loaded with the plaintext (intermediate ciphertext) sub-block X. The state of i^{th} cell y_i of the resultant configuration $Y = RCA(X, K)$ on CA rule 30 is expressed as in Equation (8)

$$y_i = {k_j}^l{}_{i-1} \oplus ({k_j}^l{}_i \vee {k_j}^l{}_{i+1}) \oplus x_i. \quad (8)$$

RS: The Reversible-Substitution (RS) operations substitute byte by byte (or word by word) using self invertible operations. The objective of two different RS (i.e. RS1 and RS2) in encryption process ensures that the differential is not zeroed out. Both these RS operations are chosen to be CA based for achieving high parallelism. The periodic boundary CA skewed_rule 45 and skewed_rule 30 are chosen. The state of i^{th} cell x_i evolved using RS1 and RS2 are respectively shown in Equations (4) and (5).

BP: The Bit-Permutation (BP) operation permutes the i^{th} bit into $((9 * i) mod\ 31 + 1)^{th}$ bit. The idea behind this permutation is to place the three neighborhood bits into three different bytes. This increases the rate of diffusion and makes differential cryptanalysis difficult. Moreover implementation of this permutation is very simple can be hard-wired simply by wire-crossings.

NCA: Diffusion-box in the LH-round of the proposed encryption/decryption algorithm uses NCA rule 218 (discussed in Section 2).

4.1.1 Operational Details

The 128-bit plaintext block P can be represented as sequence of four (32-bit word) sub-blocks $(p_1, \ldots, p_4)$. These four plaintext sub-blocks are transformed into the four ciphertext sub-blocks $(c_1, \ldots, c_4)$ under the influence of $(r * 5 + 4)$ key sub-blocks in *((r + half) rounds)* complete operation. A single complete round operation comprises of two half rounds. ζ_{FH} and ζ_{LH} represent respectively for the first-half and last-half round operations discussed as follows.

FH-round: Each FH-round operation (ζ_{FH}) accepts ${ic_1}^l, \ldots, {ic_4}^l$ and ${k_1}^l, \ldots, {k_4}^l$ 32-bit as input performs very simple RCA based operation for key-mixing followed by a (skewed_CA rule) reversible operation. The plaintext $p_1, \ldots, p_4$ is treated here as ${ic_1}^1, \ldots, {ic_4}^1$

$$\begin{aligned} {a_1}^l, {a_2}^l, {a_3}^l, {a_4}^l &= \zeta_{FH}[{ic_1}^l, {ic_2}^l, {ic_3}^l, {ic_4}^l, \\ &\qquad {k_1}^l, {k_2}^l, {k_3}^l, {k_4}^l] \\ i.e.,\ {a_1}^l &= RS1(RCA({k_1}^l, {ic_1}^l)) \\ {a_3}^{l+1} &= RS1(RCA({k_2}^l, {ic_2}^l)) \\ {a_2}^{l+1} &= RS2(RCA({k_3}^l, {ic_3}^l)) \\ {a_4}^{l+1} &= RS2(RCA({k_4}^l, {ic1_4}^l)). \end{aligned}$$

LH-round: LH-round operation (ζ_{LH}) accepts $a_1^l, \ldots, a_4^l$ (the outputs of FH-round), and k_5^l to generate the intermediate ciphertext to be used in next $(l+1)$ round.

$$[{ic_1}^{l+1}, \ldots, {ic_4}^{l+1}] = \zeta_{LH}[{a_1}^l, \ldots, {a_4}^l, {k_5}^l].$$

These intermediate cipher for next round can also be expressed as

$$\begin{aligned} &[{ic_1}^{l+1}, {ic_2}^{l+1}, {ic_3}^{l+1}, {ic_4}^{l+1}] \\ = \ &[{a_1}^l \oplus {t_4}^l, {a_2}^l \oplus {t_5}^l, {a_3}^l \oplus {t_4}^l, {a_4}^l \oplus {t_5}^l], \end{aligned}$$

where ${t_4}^l, {t_5}^l$ can be represented as

$$\begin{aligned} {t_4}^l &= NCA(BP(NCA({a_1}^l \oplus {a_2}^l, {k_5}^l), BP({a_3}^l \oplus {a_4}^l))) \\ {t_5}^l &= NCA(BP({a_3}^l \oplus {a_4}^l), BP(NCA({a_1}^l \oplus {a_2}^l, {k_5}^l))). \end{aligned}$$

The computational flow for the proposed block cipher is depicted in Figure 3 and the pseudocode for encryption

process is given below.

Pseudocode for proposed encryption scheme:
//Uppercase alphabetset represent 128-bit block and
// corresponding lowercase sets specify 32-bit words
Input: P (p1,p2,p3,p4): plaintext block, K: user selected key
Output: C (c1,c2,c3,c4): ciphertext block
IC (ic1,ic2,ic3,ic4) :intermediate cipher block
a1,..,a4, t1..t5 are 32-bit words
Ek[*l*][*j*]:j^{th} *key sub-block in* l^{th} *round.*

BEGIN
// Call subroutine to generate key sub-blocks for each round//
Keyschedule(K);
IC := P;
for l := 1 to r do
// Stage-1//
a1:=RCA(ic1,Ek[*l*][1]); a2:=RCA(ic2,Ek[*l*][2]);
a3:=RCA(ic3,Ek[*l*][3]); a4:=RCA(ic4,Ek[*l*][4]);
RS1(a1); RS1(a2); RS2(a3); RS2(a4);
//Stage-2//
t1 := a1 $\oplus$ a3; t2 := a2 $\oplus$ a4;
BP(t2); t3 := NCA(Ek[*l*][5], t1); BP(t3);
t4 := NCA(t3, t2); t5 := NCA(t2,t3);
ic1 := a1 $\oplus$ t4; ic2 := a3 $\oplus$ t4;
ic3 := a2 $\oplus$ t5; ic4 := a4 $\oplus$ t5;
endfor
// Last (r+1)th round (FH-round)//
c1:= RCA(ic1,Ek[r+1][1]); RS1(c1);
c2:= RCA(ic3,Ek[r+1][2]); RS1(c2);
c3:= RCA(ic2,Ek[r+1][3]); RS2(c3);
c4:=RCA(ic4,Ek[r+1][4]); RS2(c4);
END.

4.1.2 Why Decryption Works?

The computational flow of the decryption process is essentially same process as that of the encryption process. The only change being that the schedule of the key sub-blocks are computed for that of the encryption as presented in the key schedule. Denoting $Dk_l{}^i$ and $Ek_l{}^i$ for decryption key and encryption key for i^{th} sub-block of l^{th} round.

$$\forall 1 \le l \le r$$
$$[Dk_l{}^1, Dk_l{}^2, Dk_l{}^3, Dk_l{}^4, Dk_l{}^5] =$$
$$[Ek^1{}_{r+2-l}, Ek^3{}_{r+2-l}, Ek^2{}_{r+2-l}, Ek^4{}_{r+2-l}, Ek^5{}_{r+1-l}];$$
$$[Dk^1_{r+1}, Dk^2{}_{r+1}, Dk^3{}_{r+1}, Dk^4{}_{r+1}] =$$
$$[Ek_1{}^1, Ek_1{}^3, Ek_1{}^2, Ek_1{}^4].$$

The reason behind the same structure works for both encryption and decryption is as follows.

- Each FH-round uses a reversible key mixing operation followed by a reversible substitution operation and therefore, the effect can be cancelled by using the same key blocks in the decryption process.

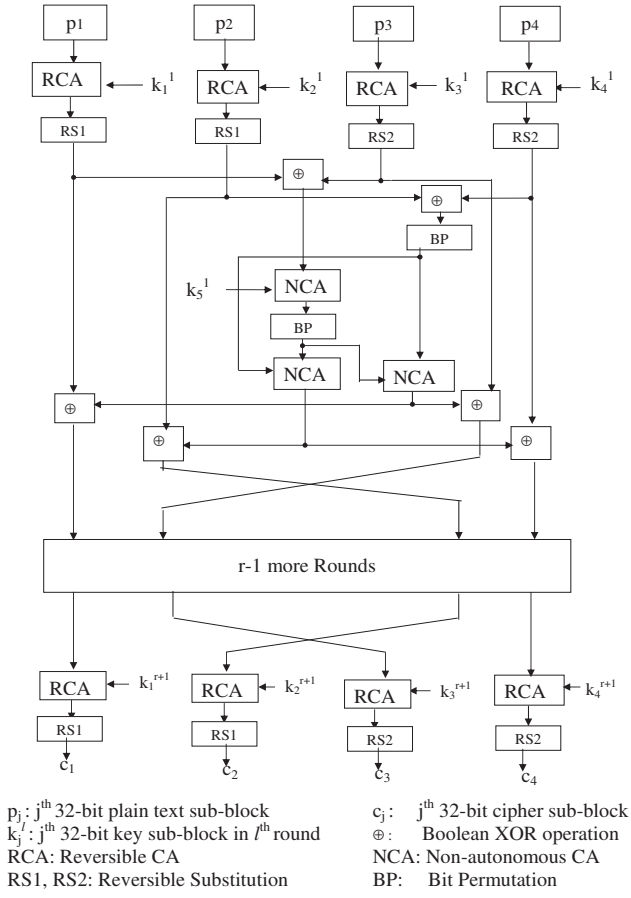

Figure 3: Computation flow for LCASE

- The structure used in the LH-round has self invertible property i.e., the input sub-blocks $a_1{}^l, \ldots, a_4{}^l$ can be obtained using the output sub-blocks $ic_1{}^{l+1}, ic_2{}^{l+1}, ic_3{}^{l+1}, ic_4{}^{l+1}$.

4.2 Key Schedule

The primary objectives of our proposed key-schedule are

- The key sub-block should be a cryptographic pseudo random, and collision resistant;
- Each round key should involve maximum number of user selected key bits;
- Ease of implementation.

The proposed key schedule uses a function called *key sub block generator* (*KSG(.)*). The function KSG(.) as depicted in Figure 4 involves 16 number of 8-cell periodic boundary RCA based on CA rule 45. The subroutine KSG(.) is illustrated in pseudocode for proposed key schedule that accepts α of size 128-bits and returns β (same size) as a result.

We describe here, the key expansion of 192-bit user selected key to generate 74 number of 32-bit key-sub blocks to be used in 14 and half rounds complete encryption process (key schedules for other 128 /256-bit user selected keys are similar to this). The bit stream of K can be treated as an array of 32 bit-words. The user selected key K is initialized to $kk[1], \ldots, kk[kw]$, where kw is the size of key in number of words (e.g. for 192-bit key kw is 6). The first 4 words of kk (*i.e.*, kk[1]...,kk[4]) are used to generate the next key sub blocks (using the subroutine *KSG(.)*)and the resulting sub-blocks are treated as kk[7]..kk[10]. Next, kk[5]..kk[8] sequence is used to generate kk[11]..kk[14] and so on. The process is iterated until kk[74] is obtained. The sequence kk[1],. . .,kk[5*r+4] can be treated as 2-dimensional encryption key $Ek[l][j]$. Here, $Ek[l][j]$ represent a 32-bit j^{th} key word of l^{th} round. The decryption keys(Dk) are computed from encrypted key sub-blocks (Ek). The pseudocode for the proposed key schedule is presented below.

Pseudocode for proposed key-schedule
Input : K (k[1]..k[kw]): key
Output: Ek[l][j],Dk[l][j] // j^{th} key sub-block for l^{th} round
r: the number of rounds, kk[t]: t^{th} 32-bit sub-block.
BEGIN
// *Encryption Key computation*//
for j := 1 to kw do kk[j] := k[j];
l:=1; j := kw + 1; t:=1;
// *Generate key sub-blocks for each round*//
while ($j \leq (r * 5 + 4)$) do
// *Call subroutine KSG* //
(kk[j]..kk[j+kw]) := KSG(kk[t]..kk[t+kw]);
t:= t + kw; j:= j + kw;
endwhile
for l:= 1 to r do
for j := 1 to 5 do Ek[l][j] := kk[t++];
for j := 1 to 4 do Ek[l][j] := kk[t++]; //Last round key//
// *Determination of Decryption Key Dk from Ek*//
for l:= 1 to r do
Dk[l][1] := Ek[r+2-l][1]; Dk[l][4]):=Ek[r+2-l][4];
Dk[l][2]:=Ek[r+2-l][3]; Dk[l][3]:=Ek[r+2-l][2];
Dk[l][5]:=Ek[r+1-l][5];
endfor
Dk[r+1][1] :=Ek[1][1]; Dk[r+1][2] := Ek[1][2];
Dk[r+1][3] := Ek[1][3]; Dk[r+1][4]):=Ek[1][4];
END.

Subroutine KSG(α)
//$\alpha(\beta)$ can be considered as $\alpha_1, \ldots, \alpha_{16}$ $(\beta_1, \ldots, \beta_{16})$ //
for i:=1 to 16 do
t:= := (i+7) mod 16 + 1;
$\beta_t := RCA(\alpha_i, \alpha_{i+1 \ mod \ 16})$;
endfor
return(β)

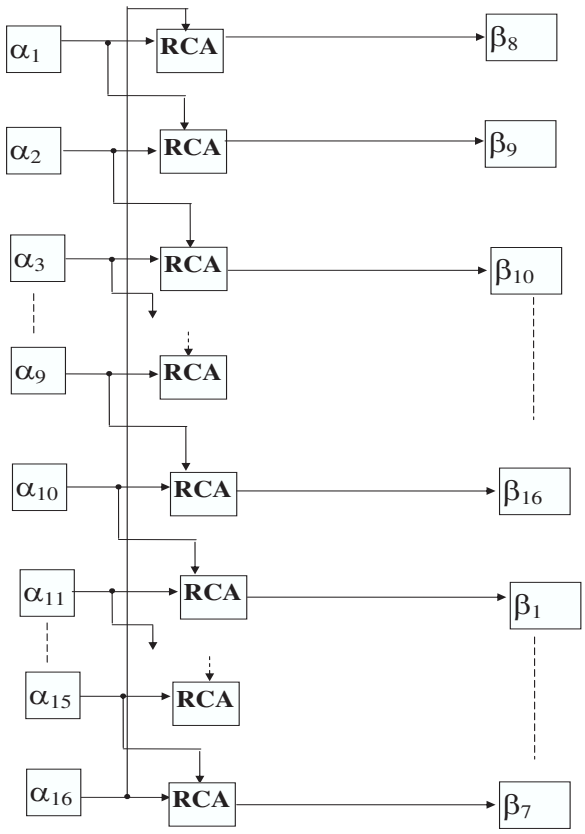

Figure 4: Key sub-block generation

5 Security Analysis

Security analysis may be treated as the process of analyzing the robustness against known attacks. However, the basic design properties are primary requirements to achieve the security.

5.1 Design Properties

Diffusion and confusion [21] are the most desirable criteria for a block cipher. The resulting design achieves them as follows:

Diffusion: The purpose of diffusion on a cipher is to spread out redundancy of plaintext all over the cipher, i.e., each bit of the plaintext needs to contribute to as many cipher bits. LCASE satisfies the diffusion requirement by the clever arrangement of 3-neighborhood CA based operations with 2 BP (Bit Permutation) operations. The FH-round of LCASE round raises the dependencies of each bit to 10 key-bits and 6 text-bits which are attributed to the 3-neighborhood RCA operation followed by RS operation. The two 32-bit XOR operations situated in parallel immediate after the FH-round operation makes double the bit dependencies. Further, the involution operation comprising of BP and NCA operations. The BP operation splits three adjacent bits into three separate bytes so that would

influence $3x3$ =)9-different bits of 3 different bytes by the application of a 3-neighborhood CA-based operation. It will expedite the rate of diffusion in which the output bit of the involution raises the dependencies of at least 61 key-bits and 36 text-bits. After completion of one LCASE round operation each text-bit depends on at least 66 key-bits and 39 text-bits. Hence, it can be observed that after successive two LCASE rounds, each output (intermediate cipher) bit depends on all the plaintext bits i.e, the LCASE achieves complete diffusion after two successive rounds.

Confusion: Confusion makes the relationship between key and statistics of the cipher bits complex and non-linear. LCASE achieves higher degree of confusion because of the following reasons.

- As discussed earlier, during a single round of LCASE, each output bit depends on at least 66 key-bits and 39 text bits of that round.
- The involution of RCA, based on CA rule 30 and the two RS operations based on skewed_rule 45 and skewed_rule 30, along with NCA, based on rule 218, results in the higher degree of non linearity.
- The inequality Expressions (9),(10) and (11) can be verified easily from the Equations (8) and (7), which makes the input-output relation more complex.

$$RCA(Z, X \oplus Y) \neq RCA(Z, X) \oplus RCA(Z, Y); \quad (9)$$
$$NCA(Z, X \oplus Y) \neq NCA(Z, X) \oplus NCA(Z, Y); \quad (10)$$
$$RCA(RCA(X, Z), RCA(Z, Y)) \neq RCA(X, Y). \quad (11)$$

5.2 Robustness against Known Attacks

In this section, security of the LCASE is measured considering its resistance against various known attacks

Differential cryptanalysis attack: Differential cryptanalysis [3, 13] is known to be an important conventional attack against block ciphers, introduced by Biham and Shamir, to attack against DES. The attack exploits the difference propagation from the plaintext to the ciphertext. These difference propagations can be used to assign probabilities to the possible keys, and then to locate the most probable one.

A block cipher is assumed to be secure against differential cryptanalysis attack (DCA), if the maximum differential probability is small enough to make DCA ineffective. To measure this probability, let us consider the 8-cell CA based RS operations used in the LCASE. These RS operations can be assumed as an (8-bit input to 8-bit output) S-boxes. Thus there are 16 S-boxes placed in parallel. The differential characteristic probability of LCASE has been calculated by using the following theorem discussed in [1].

Theorem 1. *If P_d be the maximum differential probability of all S-boxes and* D *be the minimum number of active S-boxes. Then, the maximum differential characteristic probability P is bounded by ${P_d}^D$.*

We searched for the worst case assumption from all entries in the difference distribution table for both the RS operations(RS1, RS2). It is found to have probability 2^{-2}. At the same time, difference in one input bit of an S-box reflects 6 different bytes after a complete round operation. So 6 active S-boxes would exist after a single round LCASE. Using Theorem 1, 11-round LCASE will have differential probability of $P \leq 2^{-2*6*11} = 2^{-132}$. It shows that there is no effective differential cryptanalysis attack on LCASE reduced to 11 or more rounds.

Linear cryptanalysis attack: Linear cryptanalysis [14] which is also known to be a powerful attack is a generic way to exploit the correlation between the input and output bits of each round. The probability that a linear expression holds true is the product of the linear probability bias in the active S-boxes and the number of active S-boxes.

LCASE design principle adopts the approaches to provide security against linear cryptanalysis have focused on optimizing the S-boxes (i.e., minimizing the largest bias) and finding structures to maximize the number of active S-boxes. Considering the 8-cell CA based RS operations used in LCASE as an (8-bit input to 8-bit output) S-boxes, there are 16 S-Boxes placed in parallel. An approximate linear expression whose output correlates to the output of non-linear function is calculated for Equations (4) and (5). The maximum probability value work out to be $\frac{12}{16}$ for both. The linear probability bias ϵ is the difference between $\frac{1}{2}$ and the probability for the linear expression i.e., $\mid \frac{1}{2} - \frac{3}{4} \mid = \frac{1}{4} = 2^{-2}$.

For each LCASE round the minimum number of active s-boxes for linear cryptanalysis is observed to be 6. Therefore, the correlation probability for one-round LCASE is $2^{-2*6} = 2^{-12}$. As r (= 12/14/16) such equal rounds are involved, the proposed scheme justifies the resistance against linear cryptanalysis attack.

Variants of linear and differential cryptanalysis attacks: Most often the basis of these attacks rest on the predictability of the polynomial constructed using input/output pairs. If intermediate bit in the encryption is represented by d-degree Boolean polynomial then the value of $(d+1)^{th}$ order differential would be 0.

During a single round of LCASE, each output bit involves minimum 66 key bits and 39 input text bits. Therefore, degree of polynomial of every output bit for a single round LCASE would be of 105. High order diffusion and more number of rounds pose their resistant towards known variants of linear and differential cryptanalysis (High order differential, Boomerang and rectangle attack) attacks.

Interpolation and algebraic attacks: Interpolation attack [9] exploits the weaknesses underlying algebraic properties of a cryptographic technique. Fortunately, combination of various (selected NCA, RCA, RS1 and RS2) operations in LCASE results in a complex expression to resist interpolation and algebraic attacks.

Related key attacks: The basic related key attacks based on uncommon hypothesis that attacker knows the relationship between two unknown different keys and corresponding ciphers. Biham [4] analyzed the security of DES-like cryptosystems. The proposed key schedule phase uses RCA based on CA as the basis. Because of the random output generation capability of CA rule 45 [29], it is very difficult to predict the relations between two sub keys from the two related keys. Additionally, the key-mixing operation of LCASE involves RCA operations (based on 3-neighborhood CA rule 30) that mix a single plaintext bit with 3 subkey-bits. Consequently, obtaining the actual sub-key bits (that was used before RCA), which is used to derive the next sub-keys, become harder.

This makes hard to obtain the actual sub-key bit (that used before RCA operation) which is used to derive the next sub keys. This is so because, retrieving the initial configuration from partial output, obtained by CA with rule 30, is NP-Complete [28].

Equivalent key attacks: LCASE involves all the bits of original user defined key in the key schedule. The Key schedule invokes key sub block generator function KSG(), to derive key-sub blocks which uses RCA based operations. Because of the reversibility property of RCA, no two different keys would result in the same round sub-keys. Therefore, it is not possible to find two different keys that would generate same sub keys.

Timing analysis attacks: Sometimes cryptanalyst breaks the cipher by analyzing time or power requirement [12]. This form of attack is possible if the time required for encryption depends on data. Exploiting this property the attacker could be able to predict the key with minimum number of chosen plaintexts. LCASE uses only 3-neighborhood periodic boundary CA based components. As a result, irrespective of the input and output bits time requirements for output generation at each individual component of LCASE rounds remain unchanged. Therefore, our proposed scheme is guarded against 'timing analysis' and 'related attacks'.

Comments on its feature: The scheme uses a vary structure that includes the non-linear CA-based operations and BP operations in each round. Therefore, retrieving key using the technique of inversion of CA [11] or the attack proposed in [5] will not be feasible.

The key-mixing in proposed encryption algorithm is performed through a CA-rule 30-based RCA that adds an extra complexity to determine the key. High degree of the non-linear Boolean operations describing in the round transformation resists the attacks similar to that discussed in [2].

6 Performance Analysis

LCASE can be implemented easily in both hardware and software. Each component of this cipher uses 3-neighborhood CA, whose inherent parallelism feature makes the parallel implementation natural. Simplicity and locality of the CA make it possible to build cheap and fast devices containing a large number of cells (processors) working in parallel. The performance of LCASE is compared with some of the existing schemes and analyzed in this section.

6.1 Hardware Implementation

Since the CA-based components are regular, modular and cascadeable structures the proposed scheme can be easily realized in hardware. To compare with existing schemes, we have synthesized the proposed scheme, using Verilog targeted to Xilinx Virtex-2. After synthesize, we found that LCASE needs only 454 (4-input) LUTs for its encryption round and 128 LUTs for key-schedule round as tabulated in Table 2. This requirement is lesser than ICEBERG [24] which needs 704 LUTs. However, the full rolling architecture of Cobra-H128 [22] DDP-based encryption algorithm requires more resources i.e., 2364 CLBs and 399 DFFs. Each round of AES Rijndael is even more critical as a single round there needs 2608 LUT and its key round needs 768 LUTs [23].

Table 2: Comparison of hardware complexity

Block Cipher	Hardware complexity (in number of 4 input LUTs)
LCASE	582 LUTs
ICEBERG	704 LUTs
AES	3376 LUTs

6.2 Software Implementation

The ICEBERG scheme [24]that proposed with the objective for efficient hardware implementation, was not efficient in software implementation. However, LCASE can be implemented quite efficiently in a wide range of processors by realizing bit-permutation (BP) as wire-crossing. To compare the execution speed of our proposal against AES, we ran the AES optimized-code [7] and our non-optimized code on a Pentium-IV 3.2GHZ processor, in windows microsoft visual C++ platform. The results are tabulated in Table 3.

Implementation speed of our scheme (non-optimized code) was found to be faster than AES (optimized code) [7] for all key sizes. This could be possible due to the inherited parallelism feature of CA. The object-code size of

Table 3: Comparison of execution time for a 128-bit block encryption

Key Size	Proposed Scheme (Non-optimized code)	AES (Optimized code)
128-bit	0.78 μs	1.09 μs
192-bit	0.93 μs	1.25 μs
256-bit	1.08 μs	1.4 μs

the proposed encryption scheme (including key-schedule phase) is 7 KB, which is significantly smaller than that for AES, i.e. 20 KB.

6.3 Suitability for Low Power Devices

LCASE is well suited to implement in the resource constrained devices.These devices have low processing power, restricted memory and limited battery power. Here, we did concentrate on 8-bit processors typical for 'smart cards' and 'sensor motes'. The RS operations in the LCASE round use 8-bit CA skewed_rule and hence can be implemented efficiently in low power devices. The 32-bit RCA(and NCA) operations of LCASE round can be substituted by four 8-bit RCA(and NCA) operations in parallel. However, the security of LCASE based on 8-bit RS operations remains unaffected. Moreover, rate of diffusion will not be degraded substantially because of the BP operations.

The key schedule can be efficiently implemented in low power devices as the process involves only 8-bit RCA based operations. Implementing the key expansion in a single shot operation is likely to occupy too much memory. The storage requirement can be minimized significantly through Dynamic key schedule. The proposed key schedule depends on the previous round key. Therefore, 128-bit/ 256-bit/ 512-bit storage for key would be sufficient to run the encryption process. At the same time, for decryption, the memory requirement would be 256-bit/ 512-bit/ 1024-bit being attributed to the second order reversible property of RCA.

This scheme uses same structure for both encryption and decryption. Also, both the procedures work with the same performance unlike the AES-Rijndael [8], where the later scheme degrades its performance into a greater extent in the decryption phase. LCASE is well suited for dedicated hardware implementation that substantially reduces the code size. All constructions are efficient for implementation, especially on low-end devices, which need a small memory and less computation.

7 Conclusion

In this paper, we have proposed a light weight symmetric key cryptosystem using CA, called LCASE. LCASE meets the same specification as AES, that of satisfying the base security criteria (confusion and diffusion). The most advantageous features of LCASE include its simple realization, small code size, low memory requirement and faster implementation. The proposed encryption algorithm is resistant against most generous attacks such as differential cryptanalysis and linear cryptanalysis attacks while the key schedule is secure against related key and equivalent key attacks. LCASE is robust against timing analysis attacks too.

References

[1] K. Aoki, T. Ichikawa,M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms-design and analysis," *SAC-2000*, LNCS 2012, pp. 39-56, Springer-Verlag, 2001.

[2] F. Bao, "Cryptanalysys of a partially known cellular automata cryptosystem," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1493-1497, 2004.

[3] E. Biham, and A. Shamir, "Differential cryptanalysis of DES like cyptosystems," *Journal of Cryptology*, vol. 4, pp. 3-72, 1991.

[4] E. Biham, 'New types of cryptanalytic attacks using related keys," *Eurocrypt '93*, LNCS 765, pp. 229-248, Springer-Verlag, 1994.

[5] S. Blackburn, S. Murphy, and K. Paterson, "Comments on theory and application of cellular automata in cryptography," *IEEE Transactions on Computers*, vol. 46, no. 5, pp. 637-638, 1997.

[6] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications*, Wiley-IEEE Computer Society Press, Vol. 1, 1997.

[7] Christophe Devine, Free Software Implementation. (http://www.cs.rochester.edu/ brown/Crypto/src/ AES.c)

[8] J. Daemen, and V. Rijmen, *Specification for the Advanced Encryption Standard (AES)*, Springer-Verlag, 2002.

[9] T. Jakobson, and L. Knudsen, "The interpolation attacks on block ciphers," *FSE-97*, LNCS 1267, pp. 28-40, Springer-Verlag, 1997.

[10] J. Kelsey, B. Schneier, and B. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES," *Crypto '96*, LNCS 1109, pp. 237-251, Springer-Verlag, 1997.

[11] C. K. Koc, and A. M. Apohan, "Inversion of cellular automata iteration," *IEE Proceedings of Computer and Digital Technique*, vol. 144, no. 5, pp. 279-284, 1997.

[12] P. Kocher, J. Jaffre, and B. Jun, "Differential power analysis," *Crypto'99*, LNCS 1666, pp. 398-412, Springer-Verlag, 1999.

[13] X. Lai, and J. L. Massey, "Markov ciphers and differential cryptanalysis," *Crypto '91*, LNCS 576, pp. 17-38, Springer-Verlag, 1992.

[14] M. Matsui, "Linear cryptanalysis method for DES cipher," *Eurocrypt '93*, LNCS 765, pp.386-397, Springer-Verlag, 1994.
[15] A. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Oct. 1996.
[16] N. A. Moldovyan, P. A. Moldovyan, and D.H. Summerville, "On software implementation of fast DDP-based ciphers," *Internatiol Journal of Network Security*, vol. 4, no. 1, pp. 81-89, 2007.
[17] S. Nandi, B. K. Kar, and P. P. Chaudhuri, "Theory and application of cellular automata in cryptography," *IEEE Transaction on Computers*, vol. 43, no. 12, pp. 1346-1357, 1994.
[18] S. Sen, C. Shaw, D. R. Chowdhuri, N. Ganguly, and P. Pal Chaudhuri, "Cellular automata based cryptosystem (CAC)", *ACRI-2002*, LNCS 2513, pp. 303-314, Springer-Verlag, 2002.
[19] F. Seredynski, K. Pienkosz, and P. Bouvry, "Reversible cellular automata based encryption," *NPC '04*, LNCS 3222, pp. 411-418, Springer-Verlag, 2004.
[20] F. Seredynski, P. Bouvry, and A. Y. Zomaya, "Cellular automata computation and secret key cryptography," *Parallel Computing*, vol. 30, pp. 753-766, 2004.
[21] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 656-715, 1949.
[22] N. Sklavos, N. A. Moldovyan, and O. Koufopavlou, "High speed networking: Design and implementation of two new DDP-based ciphers," *Mobile Networks and Applications-MONET*, vol. 25, no. 1-2, pp. 219-231, Springer-Verlag, 2005.
[23] F. Standaert, G. Rouvroy, J. Quisquater and J. Legat, "A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES rijndael," *FPGA-2003*, pp. 281-291, California, 2003.
[24] F. Standaert, G. Piret, G. Rouvroy, J. Quisquater, and J. Legat, "ICEBERG : An involutional cipher efficient for block encryption in reconfigurable hardware," *FSE '04*, LNCS 3017, pp. 279-299, Springer-Verlag, 2004.
[25] T. Toffoli, and N. Margolus, "Invertible cellular automat: A review," *Physica D*, vol. 45, pp. 229-253, (reprinted with correction as of Oct. 2001).
[26] S. Tripathy, and S. Nandi, "Cryptosystem for low-power devices", *Turbocoding-2006*, Munich, Germany, Apr. 2006.
[27] S. Wolfram, "Cryptography with Cellular Automata," *Crypto '85*, LNCS 218, pp. 429-432, Springer-Verlag, 1986.
[28] S. Wolfram, "Random sequence generation by cellular automata," *Advances in Applied Maths*, vol. 7, no. 2, pp. 123-169, 1986.
[29] S. Wolfram, *A New kind of Science*, Wolfram media Inc. 2002.

Somanath Tripathy has received his M. Tech (C.S.T) degree in 2000 from Andhra University and Ph D degree in Computer Science and Engineering in 2007 from Indian Institute of Technology Guwahati. He was a Reader in Department of Information Technology of Northeastern Hill University, Shillong during 2006-2008. Recently, he has joined as Asst. Professor in Indian Institute of Information Technology and Management, Gwalior. His research interest includes information security and cryptography especially for resource constrained devices. He has published 16 International Journals/ Conferences papers.

Sukumar Nandi received B Sc (Physics), B Tech and M Tech from Calcutta University in 1984, 1987 and 1989 respectively. He received the Ph D degree in Computer Science and Engineering from Indian Institute of Technology Kharagpur in 1995. In 1989-90 he was a faculty in Birla Institute of Technology, Mesra, Ranchi, India. During 1991 to 1995, he was a scientific officer in Computer Sc & Engg, Indian Institute of Technology Kharagpur. In 1995 he joined in Indian Institute of Technology Guwahati as an Assistant Professor in Computer Science and Engineering. Subsequently, he became Associate Professor in 1998 and Professor in 2002. He was in School of Computer Engineering, Nanyang Technological University, Singapore as Visiting Senior Fellow for one year (2002-2003). He was General Vice-Chair of 8th International Conference on Distributed Computing and Networking 2006. He was General Co-Chair of the 15th International Conference on Advance Computing and Communication 2007. He is also involved in several international conferences as member of advisory board/ Technical Programme Committee. He is reviewer of several international journals and conferences. He is co-author of a book titled "Theory and Application of Cellular Automata" published by IEEE Computer Society. He has published more than 150 Journals/Conferences papers. His research interests are Computer Networks (Traffic Engineering, Wireless Networks), Computer and Network security and Data mining. He is Senior Member of IEEE and Fellow of the Institution of Engineers (India).