Rain: Reversible Addition with Increased Nonlinearity

Jaydeb Bhaumik[1], Debdeep Mukhopadhyay[2], and Dipanwita Roy Chowdhury[2]
(Corresponding author: Jaydeb Bhaumik)
Dept. of ECE, Haldia Institute of Technology, Haldia, West Bengal, India, Pin-721657[1]
Dept. of CSE, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India, Pin-721302[2]
(Email: bhaumik.jayeb@gmail.com)
(Received Oct. 12, 2010; revised and accepted Mar. 28, 2011)

Abstract

This paper proposes a reversible, balanced and nonlinear vectorial Boolean function called 'Rain'. Traditional integer addition modulo 2^n has several features like reversibility, balancedness and nonlinearity. However, the bias for the best linear approximation of the output bits and their linear combinations is quite high. This leads to several attacks on stream cipher like NLS, which employs addition modulo 2^n. In this paper, it has been proved mathematically that the bias of the each output bit and their non-zero linear combinations of the proposed vectorial Boolean function decreases exponentially with the bit position. Also as a case study, it has been shown that attacks against stream cipher NLS is prevented through the incorporation of Rain.

Keywords: Boolean function, NLS and crossword puzzle attack, nonlinearity, reversibility

1 Introduction

Boolean functions play an important role in the design of symmetric key cryptographic algorithms. In case of block ciphers like AES and DES, nonlinearity is provided by the substitution boxes (S-box) solely. S-boxes are the most important and complex part of many block ciphers. Implementation of S-box in software or hardware requires either significant amount of memory or silicon area. Therefore, S-boxes make the block ciphers unsuitable for light weight cryptography [6]. Also, recent findings show that look up table based implementation of S-boxes are prone to cache-timing and other side channel attacks [4, 5]. Thus it may be prudent at this point to look into Boolean circuits which provide high nonlinearity.

Addition modulo 2^n has wide range of applications in the design of many cryptographic primitives like stream cipher, block cipher and hash function, because it is nonlinear, reversible and balanced function. Stream ciphers like HC-128, Rabbit, Salsa20, NLS and Helix employ addition modulo 2^n for key stream generation. In case of block ciphers like IDEA, MARS, FEAL, SEA addition *mod* 2^n is used for round key mixing operation. Secure hash algorithms SHA-0 and SHA-1 also employ addition modulo 2^n to compute hash value. Although addition modulo 2^n has wide range of applications but bias of the best linear approximation of the output bit is quite high. The bias of the best linear approximation for the output bit of addition *mod* 2^n has been studied in [9, 14]

Cho and Pieprzyk presented a linear distinguishing attack called Crossword Puzzle Attack (CPA)[2] on the stream cipher NLS [10], which has been submitted to eSTREAM project. In [8], MacDoland and Hawkes have proposed an improved version of CPA on NLS by exploiting the internal dependencies between NFSR and NLF. Linear approximations based on adjacent bits are used to build a distinguisher for the case $Konst = 0$ with bias $2^{-19.7}$ [8]. CPA [2] has been claimed to be prevented in [1] using a vectorial Boolean function *Slash* instead of addition modulo 2^n in the nonlinear filter (NLF) of NLS. The function *Slash* [1] is nonlinear, reversible and has a strong resistance against linear cryptanalysis. Also it has been shown that hardware implementation cost and time delay of *Slash* is less compared to addition *mod* 2^n. However like modular addition, *Slash* has the demerit that the bias [7] of the XOR of consecutive output bit positions is held constant at $\frac{1}{4}$. It is shown that NLS can still be attacked by the CPA in spite of *Slash*, because of the property stated above which makes *Slash* virtually equivalent to modular addition. This motivates the study of highly nonlinear, reversible and balanced vectorial Boolean function.

In this paper, a new vectorial Boolean function called 'Rain' (**R**eversible **A**ddition with **I**ncreased **N**onlinearity) has been proposed which is nonlinear, balanced and also reversible. The nonlinearity of each component function has been computed using basic definition of nonlinearity [13]. It provides better resistance against linear cryptanalysis compared to addition and *Slash*. It has been shown that the bias of the XOR of consecutive bits position in the output also decreases exponentially with bit

position. Such a strong reversible function may be employed to plague cryptanalysis like CPA against NLS cipher on dint of the property that XOR of successive terms have a low bias compared to either modulo addition in the original NLS [10] proposal or *Slash* function suggested in the protection scheme [1].

The rest of the paper is organized as follows. Section 2 discusses some preliminaries required for this work. The function Rain is elaborated in section 3. Performance of the proposed function against linear cryptanalysis has been discussed in section 4. In section 5, hardware and time complexity of the scheme is given. Section 6 introduces one application of the proposed Boolean function and section 7 concludes the work.

2 Preliminaries

Some basic definitions and notations are discussed in this section.

Definition 1. *A function $y = \xi(x)$ that maps a Boolean vector to another Boolean vector is called a vectorial Boolean function, where y is an m-bit output Boolean vector and x is an n-bit input Boolean vector.*

Definition 2. *A Boolean function $\xi(x) : Z_2^n \rightarrow Z_2$ is a mapping from n-bit inputs to one bit output.*

Definition 3. *A Boolean function $\xi(x)$, where x is an n-bit input Boolean vector, can be uniquely written as a sum (XOR) of products (AND). This is known as Algebraic Normal Form (ANF).*
$\xi(x_1, x_2, ..., x_n) = p_0 \oplus p_1x_1 \oplus p_2x_2 \oplus p_nx_n \oplus p_{12}x_1x_2 \oplus ... \oplus p_{12...n}x_1x_2...x_n$, *where* $p_0, p_1, ..., p_{12...n} \in \{0, 1\}$.

Definition 4. *The Hamming weight of a binary string x is the number of 1's in the string and it is denoted by $wt(x)$.*

Definition 5. *The Hamming distance between two binary strings (say x and y) of equal length is measured by $wt(x \oplus y)$.*

Definition 6. *An n variables Boolean function $\xi(x_1, x_2, ..., x_n)$ is said to be an affine function if the ANF of ξ is of the form $\xi(x_1, x_2, ..., x_n) = p_0 \oplus p_1x_1 \oplus p_2x_2 \oplus ... \oplus p_nx_n$, where $p_0, p_1, ...p_n \in \{1, 0\}$. If p_0 is 0 then the function is said to be linear.*

Definition 7. *Nonlinearity of an n variable Boolean function ξ is defined as the minimum Hamming distance from the set of all affine function of n variables.*

Definition 8. *The bias for best linear approximation [13] is defined as $p_i - \frac{1}{2}$, where p_i is the probability of best linear approximation.*

Definition 9. *If the biases of k of independent random variables $X_{i_1}, \ldots, X_{i_k}$ are $\epsilon_{i_1}, \ldots, \epsilon_{i_k}$ respectively and $\epsilon_{i_1,i_2,...i_k}$ denotes the bias of the random variable $X_{i_1} \oplus ... \oplus X_{i_k}$. Then $\epsilon_{i_1,i_2,...i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$ [13].*

Definition 10. *A Boolean function $\xi(x)$ of n variables, where n is even, is called a Bent function if it has a nonlinearity value $2^{n-1} - 2^{n/2-1}$ [12]. This is the highest possible nonlinearity for an n variable Boolean function if n is even.*

Theorem 1. *The nonlinearity of Boolean function $f(x_n, ..., x_1) \oplus g(y_m, ..., y_1)$ is $2^n nl(g) + 2^m nl(f) - 2nl(f)nl(g)$, where $f(x_n, ..., x_1)$ and $g(y_m, ..., y_1)$ are the two boolean functions of n and m variables respectively, $\{x_n, ..., x_1\} \cap \{y_m, ..., y_1\} = \phi$ and $nl(f)$, $nl(g)$ denote the nonlinearity of f and g respectively [12].*

3 Proposed Nonlinear Function

In this section, a new nonlinear, reversible and balanced vectorial Boolean function denoted as 'Rain' and its inverse function denoted as 'I-Rain' are introduced.

Definition 11. *Assume $X = (x_{n-1}\ x_{n-2}\ \ldots\ x_0)$ and $K = (k_{n-1}\ k_{n-2}\ \ldots\ k_0)$ are two $n-$bit inputs and $Y = (y_{n-1}\ y_{n-2}\ \ldots\ y_0)$ is the $n-$bit output after mixing X with K, where x_0, k_0 and y_0 denote the LSBs and x_{n-1}, k_{n-1} and y_{n-1} denote the MSBs. The function* Rain *is denoted by the operator $\dagger$ and is defined as $Y = (X \dagger K) = \mathsf{F}(X, K)$, where*

$$y_i = x_i \oplus k_i \oplus c_{i-1}; \qquad c_i = \bigoplus_{j=0}^{i} x_j \cdot k_{i-j} \tag{1}$$

where $\oplus$ is modulo-2 sum, $\cdot$ represents AND operation, $0 \leq i \leq n-1$, $c_{-1} = 0$ and c_i is the carry term propagating from i-th bit position to $(i+1)$-th bit position. The end carry c_{n-1} is neglected.

It is noted that c_{i-1} is a Boolean function of $2i$ variables and it is in the form of bent function and obviously it is not balanced. Whereas, y_i can be considered as a combination of two functions: one linear function $x_i \oplus k_i$ and a bent function c_{i-1}. Therefore, y_i is balanced.

Definition 12. *Inverse function takes two n-bit inputs $Y = (y_{n-1}\ y_{n-2} \ldots y_0)$ and $K = (k_{n-1}\ k_{n-2} \ldots k_0)$ and produces an $n-$bit output $X = (x_{n-1}\ x_{n-2} \ldots x_0)$. Inverse function* I-Rain *is defined as $X = \mathsf{F}^{-1}(Y, K)$, where*

$$x_i = y_i \oplus k_i \oplus d_{i-1}; \qquad d_i = \bigoplus_{j=0}^{i} x_j \cdot k_{i-j} \tag{2}$$

where $\oplus$ is modulo-2 sum, $\cdot$ represents AND operation, $0 \leq i \leq n-1$, $d_{-1} = 0$ and d_i is the carry term propagating from i-th bit position to $(i+1)$-th bit position. The end carry d_{n-1} is neglected.

3.1 Analogy Between Rain and Addition

Following four properties show the similarity of Rain with integer addition.

Property 1: $\mathsf{F}(A, A) \neq 0$ if any $a_k \neq 0$, where $0 \leq k \leq \lfloor \frac{n-2}{2} \rfloor$; $\mathsf{F}^{-1}(A, A) = 0$.

Analogy: $A + A = 2A \neq 0\ \forall\ A \neq 0$; $\quad A - A = 0$.

Proof. Assume $Y = \mathsf{F}(A, A)$, where A and Y are two n-bit variables. Then from the definition of F we can write

$$\begin{aligned} y_i &= a_i \oplus a_i \oplus \bigoplus_{j=0}^{i-1} a_j a_{i-1-j} \\ &= \bigoplus_{j=0}^{i-1} a_j a_{i-1-j}. \end{aligned}$$

It is observed that $y_{2k+1} = a_k$ and $y_{2k} = 0$, where $0 \leq k \leq \lfloor \frac{n-2}{2} \rfloor$ i.e. even output bits are zero. Hence $\mathsf{F}(A, A) \neq 0$ if any $a_k \neq 0$, where $0 \leq k \leq \lfloor \frac{n-2}{2} \rfloor$
Assume $X = \mathsf{F}^{-1}(A, A)$, where A and X are two n-bit variables. Then from the definition of F^{-1}, we can write

$$\begin{aligned} x_i &= a_i \oplus a_i \oplus \bigoplus_{j=0}^{i-1} x_j a_{i-1-j} \\ &= \bigoplus_{j=0}^{i-1} x_j a_{i-1-j}. \end{aligned}$$

From definition of F^{-1}, $x_0 = a_0 \oplus a_0 = 0$. Since $x_0 = 0$ therefore $x_1 = 0$. Now $x_2 = 0$, because $x_1 = 0$ and $x_0 = 0$. Similarly, it can be shown that output bit x_i will be zero if all x_js are zero, where $0 \leq j \leq (i-1)$. Therefore, $\mathsf{F}^{-1}(A, A) = 0$. □

Property 2: $\mathsf{F}(A, 0) = A$; $\mathsf{F}^{-1}(A, 0) = A$.
Analogy: $A + 0 = A$; $A - 0 = A$.

Proof. Let $Y = \mathsf{F}(A, 0)$ and $X = \mathsf{F}^{-1}(A, 0)$, where A, X and Y are three n-bit variables and $A \neq 0$. From the definition of F, it is noted that $y_i = a_i$, where $0 \leq i \leq n-1$. Therefore, $\mathsf{F}(A, 0) = A$. From the definition of F^{-1}, it is observed that $x_i = a_i$, for $0 \leq i \leq n-1$. Hence $\mathsf{F}^{-1}(A, 0) = A$. □

Property 3: F^{-1} is the inverse of F i.e. $\mathsf{F}^{-1}(\mathsf{F}(A, K), K) = \mathsf{F}(\mathsf{F}^{-1}(A, K), K) = A$.
Analogy: $(A + K) - K = (A - K) + K = A$.

Proof. Assume $R = F(A, K)$ and $S = \mathsf{F}^{-1}(F(A, K), K) = \mathsf{F}^{-1}(R, K)$, where A, K, R and S are four n-bit variables. From the definition of F, we can write

$$r_i = a_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} a_j \cdot k_{i-1-j}.$$

Since $S = \mathsf{F}^{-1}(R, K)$, therefore s_i can be written as

$$\begin{aligned} s_i &= r_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} s_j \cdot k_{i-1-j} \\ &= a_i \oplus \bigoplus_{j=0}^{i-1} (a_j \oplus s_j) \cdot k_{i-1-j}. \end{aligned}$$

Let $u_i = \bigoplus_{j=0}^{i-1} (a_j \oplus s_j) \cdot k_{i-1-j}$, hence $s_i = r_i \oplus u_i$, where $0 \leq i \leq n-1$. It is noted that s_i will be equal to a_i if $u_i = 0$ i.e. for $k_l \neq 0$ if $a_m = s_m$, where $0 \leq l, m \leq i-1$. Now from the definition of F and F^{-1}, we can write $r_0 = a_0 \oplus k_0$ or $a_0 = r_0 \oplus k_0 = s_0$ and $r_1 = a_1 \oplus k_1 \oplus a_0 k_0$ or $a_1 = r_1 \oplus k_1 \oplus a_0 k_0 = s_1$. Since $a_0 = s_0$ and $a_1 = s_1$, therefore $u_2 = 0$ i.e. $s_2 = a_2$. Now $s_3 = a_3$ because $s_2 = a_2$, $s_1 = a_1$ and $s_0 = a_0$. Similarly, it can be shown that $s_i = a_i$ for $0 \leq i \leq n-1$. Therefore, $\mathsf{F}^{-1}(\mathsf{F}(A, K), K) = A$ i.e. F^{-1} is the inverse function of F.

Let $P = \mathsf{F}^{-1}(A, K)$ and $Q = \mathsf{F}(\mathsf{F}^{-1}(A, K), K) = \mathsf{F}(P, K)$, where A, K, P and Q are four n-bit variables. From the definition of F^{-1}, we can write

$$p_i = a_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} p_j \cdot k_{i-1-j}.$$

Since $Q = \mathsf{F}(P, K)$, so from the definition of F we can write

$$\begin{aligned} q_i &= p_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} p_j \cdot k_{i-1-j} \qquad \text{or} \\ q_i &= a_i \oplus k_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} p_j \cdot k_{i-1-j} \oplus \bigoplus_{j=0}^{i-1} p_j \cdot k_{i-1-j} \\ &= a_i. \end{aligned}$$

Therefore, $q_i = a_i$ for all $0 \leq i \leq n-1$ i.e. F^{-1} is the inverse function of F. Hence it is proved that $\mathsf{F}^{-1}(\mathsf{F}(A, K), K) = \mathsf{F}(\mathsf{F}^{-1}(A, K), K) = A$. □

Property 4: F satisfies but F^{-1} does not satisfy commutative law
i.e. $\mathsf{F}(X, K) = \mathsf{F}(K, X)$; $\mathsf{F}^{-1}(Y, K) \neq \mathsf{F}^{-1}(K, Y)$.
Analogy: $X + K = K + X$; $X - K \neq K - X$.

Proof. Assume $P = \mathsf{F}(X, K)$ and $Q = \mathsf{F}(K, X)$, where X, K, P and Q are four n-bit variables. From the definition of F, p_i can be expressed as

$$\begin{aligned} p_i &= x_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} x_j \cdot k_{i-1-j} \\ &= k_i \oplus x_i \oplus \bigoplus_{j=0}^{i-1} k_j \cdot x_{i-1-j} = q_i. \end{aligned}$$

Because $\bigoplus_{j=0}^{i-1} x_j \cdot k_{i-1-j} = \bigoplus_{j=0}^{i-1} k_j \cdot x_{i-1-j}$. Therefore, F satisfies commutative law.

Let $R = \mathsf{F}^{-1}(Y, K)$ and $S = \mathsf{F}^{-1}(K, Y)$, where Y, K, R and S are four n-bit variables. From the definition of F^{-1}, it can be shown that

$$\begin{aligned} r_i &= y_i \oplus k_i \oplus \bigoplus_{j=0}^{i-1} r_j \cdot k_{i-1-j} \\ &\neq k_i \oplus y_i \oplus \bigoplus_{j=0}^{i-1} r_j \cdot y_{i-1-j} = s_i. \end{aligned}$$

Hence, F^{-1} does not satisfy commutative law. □

4 Performance of Rain Against Linear Cryptanalysis

In this section, performance of the proposed function against linear cryptanalysis(LC) is discussed. The approach in LC is to determine linear expressions of the form which have a low or high probability of occurrence. Bias of best linear approximation for the output bit y_i and their linear combinations are computed in this section.

Theorem 2. *The bias of best linear approximation for y_i of* Rain *is $2^{-(i+1)}$, where $0 \leq i < n$.*

Proof. From the definition of F, it is evident that the output $y_i = x_i \oplus k_i \oplus c_{i-1}$, where c_{i-1} is the carry input into the i-th bit position and it is the only nonlinear term in y_i. From the definition of F, c_{i-1} can be expressed as

$$c_{i-1} = x_0 k_{i-1} \oplus x_1 k_{i-2} \oplus \ \ldots \ x_{i-1} k_0. \tag{3}$$

Expression shows that c_{i-1} is a function of $2i$ variables and it is in the form of bent function [11]. Since all the x_is and k_is are independent. So nonlinearity in c_{i-1} is

$$N_i = 2^{2i-1} - 2^{i-1}. \tag{4}$$

It is noted that y_i is a combination of two functions: one linear function $(x_i \oplus k_i)$ of two variables and a bent function c_{i-1} of $2i$ variables and of nonlinearity N_i. Therefore, nonlinearity of y_i is $2^2.N_i = 2^2(2^{2i-1} - 2^{i-1}) = 2^{2i+1} - 2^{i+1}$[using Theorem 1]. Since y_i is a function of $2i + 2$ independent variables, hence the number of matches in the best linear approximation of y_i is $N_m = 2^{2i+2} - 2^{2i+1} + 2^{i+1}$. Hence, the probability of matches $p_i = \frac{N_m}{2^{2i+2}} = 1 - \frac{1}{2} + \frac{1}{2^{i+1}} = \frac{1}{2} + \frac{1}{2^{i+1}}$ and the bias of best linear approximation is $\frac{1}{2^{i+1}}$. It is noted that the bias of best linear approximation of y_i decreases exponentially with the bit position i. A comparison of linear probability bias of *addition*, *Slash* and Rain for the first six output bits is shown in Table 1. □

Table 1: Comparison of bias for best linear approximation of y_i

Output bit	Bias for best linear approximation		
	Addition	*Slash*	Rain
y_0	0.50	0.50	0.50
y_1	0.25	0.25	0.25
y_2	0.25	0.125	0.125
y_3	0.25	0.0625	0.0625
y_4	0.25	0.0313	0.0313
y_5	0.25	0.0156	0.0156

Theorem 3. *The bias of best linear approximation for $y_i \oplus y_m$ of* Rain *is $2^{-(m+1)}$, where $0 \leq i, m < n$ and $m > i$.*

Proof. The bias of best linear approximation of $y_i \oplus y_m$ is derived here, where $0 \leq i, m < n$ and $i \neq m$. From the definition of F we can write

$$y_i \oplus y_m = (x_m \oplus k_m) \oplus k_i \oplus x_i \oplus c_{i-1} \oplus c_{m-1}. \tag{5}$$

From the definition of F, $x_i \oplus k_i \oplus c_{i-1} \oplus c_{m-1}$ can be expressed as

$$\begin{aligned} & x_i \oplus k_i \oplus c_{i-1} \oplus c_{m-1} \\ = & \bigoplus_{j=0}^{i-1} x_j(k_{i-1-j} \oplus k_{m-1-j}) \oplus x_i(k_{m-1-i} \oplus 1) \\ & \oplus k_i(x_{m-1-i} \oplus 1) \oplus \bigoplus_{j=i+1; j \neq m-1-i}^{m-1} x_j k_{m-1-j} \\ = & \bigoplus_{j=0}^{m-1} z_j s_{m-1-j}, \end{aligned} \tag{6}$$

where $s_{m-1-j} = k_{i-1-j} \oplus k_{m-1-j}$ for $0 \leq j \leq i-1$, $s_{m-1-j} = k_{m-1-j} \oplus 1$ for $j = i$, $s_{m-1-j} = k_{m-1-j}$ for $i+1 \leq j \leq m-1$, $z_p = x_p \oplus 1$, for $p = m-1-i$, otherwise $z_p = x_p$. Also it may be observed from the substitution that the bits s_l for $0 \leq l \leq m-1$ are statistically independent if the bits k_l for $0 \leq l \leq m-1$ are statistically independent and are uniformly chosen. Therefore, $x_i \oplus k_i \oplus c_{i-1} \oplus c_{m-1}$ is also a bent function of $2m$ variables. So, nonlinearity of $x_i \oplus k_i \oplus c_{i-1} \oplus c_{m-1}$ is $2^{2m-1} - 2^{m-1}$. Hence, nonlinearity of $y_i \oplus y_m$ is $2^2(2^{2m-1} - 2^{m-1}) = 2^{2m+1} - 2^{m+1}$ [using Theorem 1]and it is function of $2m+2$ number of variables. Therefore, the number of matches with the best linear approximation is $2^{2m+2} - 2^{2m+1} + 2^{m+1}$ and the corresponding probability of the best linear approximation is $\frac{1}{2} + \frac{1}{2^{m+1}}$. Therefore, bias of best linear approximation is $\frac{1}{2^{m+1}}$ and thus reduces exponentially with m. □

Hence, applying this observation repeatedly we conclude that if more than two bit positions are included in the linear combination, the resultant function will have the highest nonlinearity corresponding to the greatest bit position. This shows that linearly combining with other bit positions at least does not reduce the nonlinearity of the resultant function. A corollary of theorem 2 is as follows.

Corollary: The bias of best linear approximation for the non-zero linear combination of the output bits is $2^{-(m+1)}$, where m is the largest bit position involved in the linear combination.

These results show that the strength of Rain against linear cryptanalysis is high. The bias of best linear approximation for $y_i \oplus y_{i+1}$ is obtained by substituting $m = i+1$ in the corollary and the bias value is $\frac{1}{2^{i+2}}$. Table 2 shows a comparison of bias of the best linear approximation of $y_i \oplus y_{i+1}$ for addition, *Slash* and Rain. Table 2 shows that bias of best linear approximation remains fixed at $= 0.25$ for addition and *Slash*. But bias value decreases exponentially to zero for the proposed function Rain.

Table 3: Comparison of gate count and time complexity

Transformation	Mixing Function	Area (A) XOR	OR	AND	NOT	Time (T) Complexity	A× T
Forward	XOR	n				$O(1)$	$O(n)$
	Addition	$2n-1$	$n-2$	$2n-3$		$O(n)$	$O(n^2)$
	Slash	$3(n-1)$		$n-1$		$O(n)$	$O(n^2)$
	Rain	$\frac{n(n+1)}{2}$		$\frac{n(n-1)}{2}$		$O(1)$	$O(n^2)$
Reverse	XOR	n				$O(1)$	$O(n)$
	Subtraction	$2n-1$	$n-2$	$2(n-2)$	$n-2$	$O(n)$	$O(n^2)$
	I-Slash	$3(n-1)$		$2n-3$	$2(n-1)$	$O(n)$	$O(n^2)$
	I-Rain	$\frac{n(n+1)}{2}$		$\frac{n(n-1)}{2}$		$O(n)$	$O(n^3)$

Table 2: Comparison of bias of best linear approximation for $y_i \oplus y_{i+1}$

Output bit difference	Bias of best linear approx. for Addition	Slash	Rain
$y_0 \oplus y_1$	0.25	0.25	0.25
$y_1 \oplus y_2$	0.25	0.25	0.125
$y_2 \oplus y_3$	0.25	0.25	0.0625
$y_3 \oplus y_4$	0.25	0.25	0.0313
$y_4 \oplus y_5$	0.25	0.25	0.0156

5 Hardware and Time Complexity

In this section we discuss the hardware and time complexity of the proposed mixing function for n-bit block size. It is found that y_i contains i number of AND terms and two linear terms. Therefore to implement y_i, i numbers of two input AND gates and $i+1$ number of two input XOR gates are required. So implementation of an n-bit function requires $\frac{n(n-1)}{2}$ two input AND gates and $\frac{n(n+1)}{2}$ two input XOR gates. Table 3 shows a comparison of gate counts. It is observed that area × time complexity is $O(n^2)$ for all the three nonlinear function for forward transformation. The area × time complexity I-Rain is $O(n^3)$ while complexity of *I-Slash* and subtraction modulo 2^n is $O(n^2)$.
Note: Both the encryption and decryption processes use only forward transformation for generating key stream in stream cipher NLS. It is noted that area × time complexity of I-Rain is one order higher than *I-Slash* and subtraction but in the proposed application I-Rain does not play any role. Therefore, disadvantage of I-Rain does not affect the performance of modified NLS. Application of Rain is described in the following section.

6 Application of Rain

In this section, we first give a brief description of stream cipher NLS [10], Crossword Puzzle Attack(CPA) [2] and modified CPA [8] against NLS. The weakness of modified NLS, where modulo addition is replaced by *Slash* function [1] is explained next. Finally, the proposed countermeasure is discussed and it is shown that Rain provides better security against CPA.

6.1 Brief Description of NLS Stream Cipher

NLS has two components: NFSR and NLF whose work is synchronized by a clock. In NLS, key stream generator uses NFSR whose outputs are fed to the nonlinear filter NLF that produces output key stream bits. Detailed about NLS may be found in [10]. The state of NFSR at time t is denoted by $\sigma_t = (r_t[0], ..., r_t[16])$, where $r_t[i]$ is a 32-bit word. The state is determined by 17 words or equivalently 544 bits. The transition from the state σ_t to the state σ_{t+1} is defined as follows:

1) $r_{t+1}[i] = r_t[i+1]$ for $i = 0, ..., 15$;

2) $r_{t+1}[16] = f((r_t[0] <<< 19) + (r_t[15] <<< 9) + Konst) \oplus r_t[4]$;

3) $r_t[0]$ is abandoned;

4) if $t = 0$ (modulo $f16$), $r_{t+1}[2] = r_{t+1}[2] + t$.

Here $f16$ is 65537 and + is the addition modulo 2^{32}. The $Konst$ value is a 32-bit key dependent constant. The function $f : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ is constructed using an S-box with 8-bit input and 32-bit output and defined as $f(a) = \text{S-box}(a_H) \oplus a$ where a_H is the most significant 8 bits of 32-bit word a. Each output key stream word ν_t of NLF is computed by

$$\begin{aligned} \nu_t &= NLF(\sigma_t) \\ &= (r_t[0] + r_t[16]) \oplus (r_t[1] + r_t[13]) \oplus (r_t[6] + Konst). \end{aligned} \quad (7)$$

6.2 Overview of Crossword Puzzle Attack

In CPA [2], the attacker combines the linear approximation of both the NFSR and NLF to build a linear distinguisher which has high bias value. The basic steps of the attack are as follows.

1) Find a linear approximation of the non-linear state transition function used by NFSR: $l_1(\sigma_i) = \sigma_{i+1}$ with bias of ϵ_1;

2) Find a linear approximation of the non-linear function applied by NLF: $l_2(\sigma_j) \oplus l_3(\nu_j) = 0$ with bias of ϵ_2;

3) Obtain two sets of clock I and J such that $\sum_{i\in I}(l_1(\sigma_i) \oplus \sigma_{i+1}) = \sum_{j\in J} l_2(\sigma_j)$;

4) Build a distinguisher by computing $\sum_{i\in I}(l_1(\sigma_i) \oplus \sigma_{i+1}) \oplus \sum_{j\in J}(l_2(\sigma_j) \oplus l_3(\nu_j)) = l_3(\nu_j) = 0$, which has bias of $\epsilon^{|I|}.\epsilon^{|J|}$.

6.3 Improved CPA Against NLS

An improved version of CPA on NLS by exploiting the internal dependencies between NFSR and NLF is presented by MacDoland and Hawkes in [8]. Linear approximations based on adjacent bits are used to build a distinguisher for the case $Konst = 0$ with bias $2^{-19.7}$ [8]. In [8], authors have derived the linear approximation for NLF output bits considering the dependencies that exists between the internal state registers used in the NFSR and NLF. They have shown that it is possible to distinguish the NLS stream from a random stream after approximately 2^{-40} keystream words. But improved CPA on NLSv2 is not applicable, because $Konst$ changes.

6.4 Existing Countermeasure Against CPA Using *Slash*

In [1], *Slash* function has been used to thwart CPA against NLS. In case of *Slash*, least significant bits are linear so the following equation holds with probability one.

$$(r[x] \oslash r[y])_{(0)} = r[x]_{(0)} \oplus r[y]_{(0)}. \tag{8}$$

But for all $i > 0$ the linear combination of i-th and $(i-1)$-th output bits can be expressed as

$$\begin{aligned} & (r[x] \oslash r[y])_{(i)} \oplus (r[x] \oslash r[y])_{(i-1)} \\ = \; & r[x]_{(i)} \oplus r[y]_{(i)} \oplus r[x]_{(i-1)} \oplus r[y]_{(i-1)} \\ & \oplus r[x]_{(i-1)}.r[y]_{(i-1)}. \end{aligned} \tag{9}$$

where '$\oslash$' is the *Slash* operator. The bias of best linear approximation is 2^{-2} and hence *Slash* function is equivalent to modulo addition which is used in the original NLS. Similarly from the definition of *Slash* function [1], it can be shown that

$$\begin{aligned} & (r[x] \oslash r[y])_{(i)} \oplus (r[x] \oslash r[y])_{(i-1)} \\ & \oplus (r[x] \oslash r[y])_{(i-2)} \oplus (r[x] \oslash r[y])_{(i-3)} \\ = \; & r[x]_{(i)} \oplus r[y]_{(i)} \oplus r[x]_{(i-1)} \oplus r[y]_{(i-1)} \oplus r[x]_{(i-2)} \\ & \oplus r[y]_{(i-2)} \oplus r[x]_{(i-3)} \oplus r[y]_{(i-3)} \\ & \oplus r[x]_{(i-1)}.r[y]_{(i-1)} \oplus r[x]_{(i-3)}.r[y]_{(i-3)}. \end{aligned} \tag{10}$$

The bias of the best linear approximation of equation (10) is 2^{-3}. Therefore, *Slash* function has exactly same bias value for linear approximation as that of addition [3]. So, *Slash* function can not thwart the cross word puzzle attack on NLS and the complexity of the attack for the modified NLS [1] is same as that against original NLS [3]. Thus, although the bias of each output bit of *Slash* is small compared to modulo addition but the bias value remains fixed at 0.25 when successive output bits are linearly combined.

6.5 Countermeasure Against CPA Using Rain

In this work, addition modulo 2^n in NLF of NLS is replaced by Rain while NFSR remains unchanged. Hence each modified key stream word ν'_t of NLF is obtained as

$$\begin{aligned} \nu'_t &= NLF(\sigma_t) \\ &= (r_t[0] \dagger r_t[16]) \oplus (r_t[1] \dagger r_t[13]) \oplus (r_t[6] \dagger Konst) \end{aligned} \tag{11}$$

where † is Rain operator.

Analysis of modified NLS: In this section, we show how CPA can be thwarted using Rain. In the modified version of NLS, the bias of the distinguisher decreases to such a low value that any practical attack using this linear distinguisher is impossible. Since, we have not changed the NFSR, therefore the analysis of NFSR reported in [2] is also valid for our scheme. According to the structure of the non-linear shift register, the following equation holds for the least significant bit.

$$\begin{aligned} & \alpha_{t,(0)} \oplus r_t[0]_{(13)} \oplus r_t[15]_{(23)} \oplus Konst_{(0)} \\ & \oplus r_t[4]_{(0)} \oplus r_{t+1}[16]_{(0)} = 0, \end{aligned} \tag{12}$$

where α_t is the 32-bit output of the S-box that defines the transition function f, $\alpha_{t,(i)}$ and x_i are the i-th bits of the 32-bit words α_t and x respectively. From [2], the linear approximation for NFSR when $Konst = 0$ with bias of $2^{-5.35}$is given by

$$\begin{aligned} & r_t[0]_{(10)} \oplus r_t[0]_{(6)} \oplus r_t[15]_{(20)} \oplus r_t[15]_{(16)} \oplus r_t[15]_{(15)} \\ & \oplus r_t[0]_{(13)} \oplus r_t[15]_{(23)} \oplus Konst_{(0)} \\ & \oplus r_t[4]_{(0)} \oplus r_{t+1}[16]_{(0)} = 0. \end{aligned} \tag{13}$$

Next we determine the bias of linear approximation for modified NLF. We assume initially $Konst$ is zero to make our analysis simpler. Substituting $Konst = 0$ in Equation (11), we get

$$\nu'_t = (r_t[0] \dagger r_t[16]) \oplus (r_t[1] \dagger r_t[13]) \oplus r_t[6]. \tag{14}$$

From the definition of Rain, we know that the relation between LSBs are linear so the equation $(r[x] \dagger r[y])_{(0)} = r[x]_{(0)} \oplus r[y]_{(0)}$ holds with probability one. Therefore, the following equation holds with probability one

$$\begin{aligned} \nu'_{t,(0)} &= (r_t[0]_{(0)} \oplus r_t[16]_{(0)}) \oplus (r_t[1]_{(0)} \\ & \oplus r_t[13]_{(0)}) \oplus r_t[6]_{(0)}. \end{aligned} \tag{15}$$

But for $i > 0$, all bits of Rain ($\dagger$) are nonlinear. Consider the function $(r[x] \dagger r[y])_{(i)} \oplus (r[x] \dagger r[y])_{(i-1)}$. The function has linear approximation of the following form

$$\begin{aligned} & (r[x] \dagger r[y])_{(i)} \oplus (r[x] \dagger r[y])_{(i-1)} \\ = \; & r[x]_{(i)} \oplus r[y]_{(i)} \oplus r[x]_{(i-1)} \oplus r[y]_{(i-1)}, \end{aligned} \quad (16)$$

which has the bias of $2^{-(i+1)}$ (from Theorem 3). Using the above approximation, we can determine the linear approximation of $\nu'_{t,(i)} \oplus \nu'_{t,(i-1)}$ as follows

$$\begin{aligned} & \nu'_{t,(i)} \oplus \nu'_{t,(i-1)} \\ = \; & r_t[0]_{(i)} \oplus r_t[16]_{(i)} \oplus r_t[0]_{(i-1)} \oplus r_t[16]_{(i-1)} \\ & \oplus r_t[1]_{(i)} \oplus r_t[13]_{(i)} \oplus r_t[1]_{(i-1)} \\ & \oplus r_t[13]_{(i-1)} \oplus r_t[6]_{(i)} \oplus r_t[6]_{(i-1)}, \end{aligned} \quad (17)$$

with the bias of $2.(2^{-(i+1)})^2 = 2^{-(2i+1)}$.
In case of Rain, applying approximation (17), for $i > 2$ the following expression holds

$$\begin{aligned} & \nu'_{t,(i)} \oplus \nu'_{t,(i-1)} \oplus \nu'_{t,(i-2)} \oplus \nu'_{t,(i-3)} \\ = \; & r_t[0]_{(i)} \oplus r_t[0]_{(i-1)} \oplus r_t[0]_{(i-2)} \oplus r_t[0]_{(i-3)} \\ & \oplus r_t[16]_{(i)} \oplus r_t[16]_{(i-1)} \oplus r_t[16]_{(i-2)} \oplus r_t[16]_{(i-3)} \\ & \oplus r_t[1]_{(i)} \oplus r_t[1]_{(i-1)} \oplus r_t[1]_{(i-2)} \oplus r_t[1]_{(i-3)} \\ & \oplus r_t[13]_{(i)} \oplus r_t[13]_{(i-1)} \oplus r_t[13]_{(i-2)} \oplus r_t[13]_{(i-3)} \\ & \oplus r_t[6]_{(i)} \oplus r_t[6]_{(i-1)} \oplus r_t[6]_{(i-2)} \oplus r_t[6]_{(i-3)}, \end{aligned} \quad (18)$$

with bias $2^{-(2i+1)}$ (from corollary), when $Konst = 0$. Next we compute the complexity of CPA on modified NLS. The case for $Konst = 0$ has been studied at first and then the attack has been extended to $Konst \neq 0$.

Case for Konst = 0
We consider the linear approximation of $\alpha_{t,(0)}$

$$\alpha_{t,(0)} = r_t[0]_{(12)} \oplus r_t[15]_{(22)}, \quad (19)$$

which has been reported in [2]. The bias of this linear approximation is $2^{-5.46}$. By combining Equations (12) and (19), we have the following approximation

$$\begin{aligned} & r_t[0]_{(12)} \oplus r_t[15]_{(22)} \oplus r_t[0]_{(13)} \oplus r_t[15]_{(23)} \\ & \oplus r_t[4]_{(0)} \oplus r_{t+1}[16]_{(0)}, \end{aligned} \quad (20)$$

which has same bias as Equation (19) because remaining terms are linear. Approximation (20) has been divided into two parts: the least significant bits and the other bits as

$$\begin{aligned} l_1(r_t) &= r_t[4]_{(0)} \oplus r_{t+1}[16]_{(0)} \\ l_2(r_t) &= r_t[0]_{(12)} \oplus r_t[0]_{(13)} \oplus r_t[15]_{(22)} \oplus r_t[15]_{(23)}. \end{aligned} \quad (21)$$

Since, $l_1(r_t)$ contains only least significant bit variables so approximation is true with probability one. From the expression of $l_1(r_t)$, we obtain the following system of equations.

$$\begin{aligned} l_1(r_t) &= r_t[4]_{(0)} \oplus r_{t+1}[16]_{(0)} \\ &= r_{t+4}[0]_{(0)} \oplus r_{t+17}[0]_{(0)} \\ l_1(r_{t+1}) &= r_{t+1}[4]_{(0)} \oplus r_{t+2}[16]_{(0)} \\ &= r_{t+4}[1]_{(0)} \oplus r_{t+17}[1]_{(0)} \\ l_1(r_{t+6}) &= r_{t+6}[4]_{(0)} \oplus r_{t+7}[16]_{(0)} \\ &= r_{t+4}[6]_{(0)} \oplus r_{t+17}[6]_{(0)} \\ l_1(r_{t+13}) &= r_{t+13}[4]_{(0)} \oplus r_{t+14}[16]_{(0)} \\ &= r_{t+4}[13]_{(0)} \oplus r_{t+17}[13]_{(0)} \\ l_1(r_{t+16}) &= r_{t+16}[4]_{(0)} \oplus r_{t+17}[16]_{(0)} \\ &= r_{t+4}[16]_{(0)} \oplus r_{t+17}[16]_{(0)}. \end{aligned} \quad (22)$$

Adding all the approximations of Equation (22) and applying Equation (15), we get

$$\begin{aligned} & l_1(r_t) \oplus l_1(r_{t+1}) \oplus l_1(r_{t+6}) \oplus l_1(r_{t+13}) \oplus l_1(r_{t+16}) \\ = \; & \nu'_{t+4,(0)} \oplus \nu'_{t+17,(0)}. \end{aligned} \quad (23)$$

Substituting $t = t, t+1, t+6, t+13, t+16$ in the expression of $l_2(r_t)$ and after simplifying we get the following system of equations:

$$\begin{aligned} l_2(r_t) &= r_t[0]_{(12)} \oplus r_t[0]_{(13)} \oplus r_{t+15}[0]_{(22)} \oplus r_{t+15}[0]_{(23)} \\ l_2(r_{t+1}) &= r_t[1]_{(12)} \oplus r_t[1]_{(13)} \oplus r_{t+15}[1]_{(22)} \oplus r_{t+15}[1]_{(23)} \\ l_2(r_{t+6}) &= r_t[6]_{(12)} \oplus r_t[6]_{(13)} \oplus r_{t+15}[6]_{(22)} \oplus r_{t+15}[6]_{(23)} \\ l_2(r_{t+13}) &= r_t[13]_{(12)} \oplus r_t[13]_{(13)} \oplus r_{t+15}[13]_{(22)} \oplus r_{t+15}[13]_{(23)} \\ l_2(r_{t+16}) &= r_t[16]_{(12)} \oplus r_t[16]_{(13)} \oplus r_{t+15}[16]_{(22)} \oplus r_{t+15}[16]_{(23)}. \end{aligned} \quad (24)$$

Combining the set of equations in (24) with Equation (17), we get

$$\begin{aligned} & l_2(r_t) \oplus l_2(r_{t+1}) \oplus l_2(r_{t+6}) \oplus l_2(r_{t+13}) \oplus l_2(r_{t+16}) \\ & = \nu'_{t,(12)} \oplus \nu'_{t,(13)} \oplus \nu'_{t+15,(22)} \oplus \nu'_{t+15,(23)}. \end{aligned} \quad (25)$$

Combining Equations (23) and (25) we get

$$\begin{aligned} & l_1(r_t) \oplus l_1(r_{t+1}) \oplus l_1(r_{t+6}) \oplus l_1(r_{t+13}) \oplus l_1(r_{t+16}) \\ & \oplus l_2(r_t) \oplus l_2(r_{t+1}) \oplus l_2(r_{t+6}) \oplus l_2(r_{t+13}) \oplus l_2(r_{t+16}) \\ = \; & \nu'_{t,(12)} \oplus \nu'_{t,(13)} \oplus \nu'_{t+15,(22)} \oplus \nu'_{t+15,(23)} \\ & \oplus \nu'_{t+4,(0)} \oplus \nu'_{t+17,(0)} \\ = \; & 0. \end{aligned} \quad (26)$$

The bias can be computed using the piling-up lemma. As we use the approximation (20) five times and the approximation (17) twice, therefore the bias of the approximation (26) is $2^{7-1}.(2^{-5.46})^5.(2^{-27}).(2^{-47}) = 2^{-95.3}$ Therefore, complexity of the attack for the modified NLS is $2^{190.6}$. Since, the specification of the NLS cipher allows the adversary to observe up to 2^{80} keystream words per key/nonce pair [10], the attack is not successful for the modified NLS as bias of the distinguisher is less than 2^{-40}.

Case for $Konst \neq 0$
Biases of linear approximations for $\alpha_{t,(0)}$ and NLF decreases with non-zero $Konst$ and it has been explored

in [2]. According to [2], the $Konst$ has been divided into two parts as $Konst = (Konst_{(H)}, Konst_{(L)})$. Here we have explored only the case where $Konst_{(H)} \neq 0$ and $Konst_{(L)} = 0$. It has been reported in [2] that the average bias of the linear approximation for (19) is $2^{-6.19}$. In case of modified NLS, combining approximation (19) with bias $2^{-6.19}$ and the approximation (17), the bias of distinguisher (26) becomes $2^6.(2^{-6.19})^5.(2^{-27}).(2^{-47}) = 2^{-98.95}$, which is low enough to thwart any linear distinguishing attack.

Therefore, It is possible to thwart the crossword puzzle attack [2] against modified NLS, where addition modulo 2^{32} is replaced by Rain. Moreover, the modified NLS can thwart the improved crossword puzzle attack [8]. The attack proposed in [8], exploits the significantly higher bias for linear approximation of the NFSR and NLF using linear combination of adjacent bits. In NLS, only modular addition is used to provide nonlinearity in both NFSR and NLF. It is shown in [8] that bias for linear approximation of adjacent bits of input and output variables is significantly high. But in case of Rain, it is proved in Section 4 [$Theorem3$] that bias of best linear approximation for $y_i \oplus y_m$ is $2^{-(m+1)}$, where $0 \leq i, m \leq (n-1)$ and $m > i$. Also the bias of best linear approximation for the non-zero linear combination of the output bits is $2^{-(m+1)}$, where m is the highest bit position involved in the linear combination. Hence, modified CPA [8] will not succeed against modified NLS.

7 Conclusion

In this paper, a new nonlinear, balanced and reversible vectorial Boolean function called 'Rain' has been proposed. It has been shown that the nonlinear property of the Rain improves the resistance against linear cryptanalysis. Resistance of stream cipher NLS against crossword puzzle attack has been improved by replacing the modulo addition by Rain in NLF of NLS. It is shown that modified NLS can thwart the CPA. Also, the proposed function has low bias of best linear approximation for the non-zero linear combination of the output bits. Therefore, improved CPA unlikely to succeed against modified NLS.

References

[1] D. Bhattacharya, D. Mukhopadhyay, D. Saha, and D. Roy Chowdhury, "Strengthening nls against crossword puzzle attack," in *Proceedings of The 12th Australasian Conference on Information Security and Privacy*, vol. 4586, pp. 29–44, Townsville, Australia, July 2007.

[2] J. Y. Cho and J. Pieprzyk, "Crossword puzzle attack on nls," in *Proceedings of The 13th Annual Workshop on Selected Areas in Cryptography*, vol. 4356, pp. 249–265, Montreal, Canada, Aug. 2006.

[3] J. Y. Cho and J. Pieprzyk, "Linear distinguishing attack on nls," in *Proceedings of The SASC Workshop-Stream Ciphers Revisited*, Leuven,Belgium, Feb. 2006.

[4] P. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss,and other systems," in *Proceedings of Advances in Cryptology-CRYPTO*, vol. 1109, pp. 104–130, Santa Barbara, California, USA, Aug. 1996.

[5] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of Advances in Cryptology-CRYPTO*, vol. 1666, pp. 388–397, Santa Barbara, California, USA, Aug. 1999.

[6] W. K. Koo, H. Lee, Y. H. Kim, and D. H. Lee, "Implementation and analysis of new lightweight cryptographic algorithm suitable for wireless sensor networks," in *Proceedings of The International Conference on Information Security and Assurance (ISA 2008)*, pp. 73–76, Korea Univ., Seoul, Apr. 2008.

[7] M. Matsui, "Linear cryptanalysis method for des ciphers," in *Proceedings of Eurocrypt*, vol. 765, pp. 386–397, Lofthus, Norway, May 1993.

[8] C. McDonald and P. Hawkes. "On exploiting adjacent bits in nls,". tech. rep., 2006.

[9] D. Mukhopadhyay. "Design and analysis of cellular automata based cryptographic algorithms,". Tech. Rep. Ph.D thesis, I.I.T Kharagpur,India, Mar. 2007.

[10] G. Rose, P. Hawkes, M. Paddon, and M.W. de Vries. "Primitive specification for nls, 2005. http://www.ecrypt.eu.org/stream/nls.html,". tech. rep.

[11] O. S. Rothaus, "On bent functions," *Journal of Combinatorial Theory*, vol. 20, no. A, pp. 300–305, 1976.

[12] P. Sarkar and S. Mitra, "Construction of nonlinear resilient boolean functions using small affine functions," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2185–2193, 2004.

[13] D. R. Stinson, *Cryptography Theory and Practice*. FL, USA: CRC Press, 1995.

[14] J. Wallen, "Linear approximations of addition modulo 2^n," in *Proceedings of The 10th International Workshop on Fast Software Encryption*, vol. 2887, pp. 261–273, Lund, Sweden, Feb. 2003.

Jaydeb Bhaumik is currently working as an Associate Professor in the Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, India. During the time of doing the presented research in the paper, he was pursuing his PhD from G. S. Sanyal School of Telecommunications, Indian Institute of Technology Kharagpur, India. He received his B. Tech. and M. Tech. degrees in Radio Physics and Electronics from University of Calcutta in 1996 and 1999 respectively, and PhD degree from G. S. Sanyal School of Telecommunications, Indian Institute of Technology Kharagpur in 2010. His research interests include Cryptography, Cellular Automata, Error Correcting Codes, and Digital VLSI Design.

Debdeep Mukhopadhyay is presently working as an Assistant Professor in the Computer Science and

Engineering department of Indian Institute of Technology (IIT) Kharagpur. Prior to this he worked as an Assistant Professor in the Computer Science department of IIT Madras. He obtained his B.Tech. degree in Electrical Engineering, MS and PhD in Computer Science, IIT Kharagpur. He has been the author of more than 60 international conference and journal papers in Cryptography, Security and VLSI and has co-authored a text book on Cryptography and Network Security. He has collaborated with several organizations, like ISRO, DIT, ITI, CAIR-DRDO, Broadcom USA and NTT-Labs Japan. He has been the recipient of the Indian Semiconductor Association (ISA) TechnoInventor Award for best PhD Thesis in 2008, and the Indian National Science Academy Young Scientist Award 2010, Indian National Academy of Engineers (INAE) Young Engineer Award 2010, and the Out-standing Young Faculty Award in 2011. His research interests include Cryptography, VLSI of Cryptographic algorithms and Side Channel Analysis.

Dipanwita Roy Chowdhury is presently working as a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. She received her B.Tech. and M.Tech. degrees in Computer Science from University of Calcutta in 1987 and 1989 respectively, and PhD degree from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur in 1994. Her current research interests are in the field of Cryptography, Error Correcting Codes, Cellular Automata, and VLSI Design and Testing. She has published more than 130 technical papers in International Journals and Conferences. Also, she is a co-author of the book on "Additive cellular automata: theory and applications". She is the recipient of INSA Young Scientist Award, Associate of Indian Academy of Science and is the fellow of Indian National Academy of Engineers (INAE).