IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2019, accepted November 24, 2019, date of publication November 27, 2019,
date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956160

MRC4: A Modified RC4 Algorithm Using Symmetric Random Function Generator for Improved Cryptographic Features

RAHUL SAHA[1,2], (Member, IEEE), G. GEETHA[2,3], GULSHAN KUMAR[1,2], TAI-HOON KIM[4], AND WILLIAM J. BUCHANAN[5]

[1]School of Computer Science and Engineering, Lovely Professional University, Phagwara 144 411, India
[2]Division of Research and Development, Lovely Professional University, Phagwara 144 411, India
[3]School of Computer Applications, Lovely Professional University, Phagwara 144 411, India
[4]Beijing Jiaotong University, Beijing 100044, China
[5]Blockpass ID Lab, Edinburgh Napier University, Edinburgh EH11 4DY, U.K.

Corresponding authors: G. Geetha (gitaskumar@yahoo.com) and Tai-Hoon Kim (taihoonn@daumn.net)

ABSTRACT The Rivest Cipher 4 (RC4) has been one of the most popular stream ciphers for providing symmetric key encryption, and is now proposed as an efficient cipher within light-weight cryptography. As an algorithm it has been considered to be one of the fastest stream ciphers and one of the easiest to implement. Unfortunately, despite its simplicity of usage, a number of attacks on it have been found. Therefore, various improvements of this algorithm exist in cryptography, but none of them use proper randomness. This paper outlines modified version of RC4 and which has the desirable features of an efficient stream cipher algorithm, and which integrates the Symmetric Random Function Generator (SRFG) method. Though RC4 uses pseudorandom features with an initialisation vector and a seed value, the use of true randomness in RC4 is novel in this domain. Therefore, this paper proposes a modified RC4 as *MRC4*, and which then evaluates the statistical features of MRC4 based upon parameters such as non-linearity, resiliency, balancedness, propagation and immunity. Further, we have compared the security features and confusion-diffusion attributes with some recent variants of RC4 and have found that MRC4 is efficient in withstanding against attacks. The experimental results show that MRC4 supports a 60% better confusion property and 50% better diffusion as compared to the original RC4 method.

INDEX TERMS RC4, random number, security, cipher.

I. INTRODUCTION

Cryptography provides a fundamental security layer for data and secure services [1]. With the rise of IoT, we need improved cryptographic algorithms which are efficient in terms of security and computational complexity [2]–[4]. Designing such algorithms thus requires a number of considerations, such as key size, message size, functionality, and randomness. A key management system includes its selection, usage and background algorithm as one of the critical factors in cryptography. A weak key, or the weak algorithm with a strong key, can cause an exploitation of the ciphertext, and thus reveal the plaintext.

Along with the development of the ciphers, cryptanalysis is attractive to cryptography designers as it can identify vulnerabilities in the ciphers for further improvement. The structural design of cryptography algorithms and their corresponding technical functions are thus important considerations [5]. Logic gates including AND, OR, NOT, XOR, and XNOR are used often in the cryptographic functions of the algorithms. All these functions are good enough of developing any cryptographic algorithm, but as per the present need of randomness features, they often lag behind in exhibiting this randomness in their data processing. Moreover, functions used in cryptography possess some important features like balancedness, non-linearity, resiliency, immunity, correlation and propagation which are to be considered for evaluating the strength of the ciphers and not been evaluated significantly while designing the stream ciphers like RC4. In this paper, we have considered such parameters for RC4 in the integration of functional randomness. The cryptographic feature analysis and the inclusion of true randomness are the major contributions in this paper. For this the paper outlines a key scheduling module for MRC4 and which supports a

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

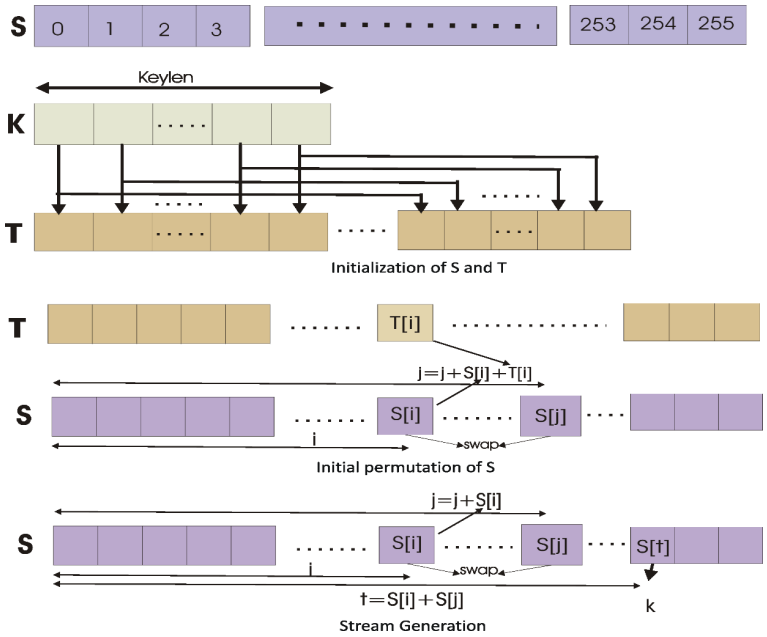

FIGURE 1. Working principle of RC4.

key scheduling modification using the Symmetric Random Function Generator (SRFG) method [6].

Section II briefly explains the basic functioning of the original RC4 algorithm and summarizes the various attacks on the algorithm. Section III shows the proposed modification in RC4 and explains its cryptographic and randomness features. Section IV covers a security analysis of MRC4 and Section V compares the performance results. Section VI finished with the core conclusions of the work.

II. THE RC4 METHOD

RC4 is a popular and widely accepted stream cipher designed in 1987 by Ron Rivest [1]. Within light-weight cryptography, such with embedded devices, RC4 is still seen to be a powerful stream cipher, especially in terms of its memory footprint, its flexible key size, its energy consumption, and its CPU utilization [29], [30]. Overall, it uses a key-size with variable length and applies byte-oriented operations with random permutation in order to create a pseudorandom stream. The security base of the algorithm depends on a pseudorandom key scheduling process, with a variable key length of between one to 256 bytes (8 bits to 2,048 bits). This is used to initialize the state vector S. For each encryption/decryption, a byte k is generated that works as a keystream part. This byte is generated by a systematic method from S. Finally, as with most stream ciphers, an XOR operation is used between the plaintext byte and the k byte. The algorithm of the original RC4, and its key scheduling, are illustrated in Figure 1.

The RC4 stream cipher has been used extensively within network protocols such as SSL, WEP, TLS, and WPA. Though this algorithm has been used widely, RC4 always has faced cryptanalysis attacks due to its several drawbacks [7], and which has reduced its adoption. Within WEP, the size of the original key in RC4 was limited to 40 bits, and which used a 24-bit Initialization Vector (IV). The small size of the key made it vulnerable to brute force attack, and the small size of the IV caused the same IV value to be rolled-over after a relatively short time. When this roll-over happened, it was then possible to reveal the plaintext [31].

In [8], the authors defined a family of related keys for each 2048-bit key, and which differs in one of the byte positions. The keystream generated by RC4 for a key and its related keys are substantially similar. Therefore, it is relatively easy to use a statistical analysis or linear and differential cryptanalysis to achieve the secret key. A statistical analysis of the RC4 keystream generator is also shown in [9], and where their process uses only $2^{30.6}$ bytes of its outputs and also analyses the distinguisher of 8-bit randomness. Researchers have also identified a number of weak keys against RC4 and which signifies that RC4 is no longer accepted for security processes [10]. Weak keys are used to generate the distinguisher for RC4 and to execute a related key attack on the cipher. It is also shown that a proposed passive ciphertext attack procedure can break any arbitrary long key within practical time complexities. The authors also analysed the Fortuitous states in RC4.

A statistical bias has been identified in the distribution of the first two output bytes of the RC4 keystream generator [11]. This work shows that 2^{25} bytes of outputs are sufficient to effectively measure the RC4 outputs from random strings. A cryptanalytic attack that uses the tree representation of RC4 cipher is also analysed [12]. It introduces an abstraction for managing the information about its internal state. A hill-climbing strategy is then used to find out the initial state. The simple complexity of this attack as compared to exhaustive search confirms that RC4 is weak. This attack is

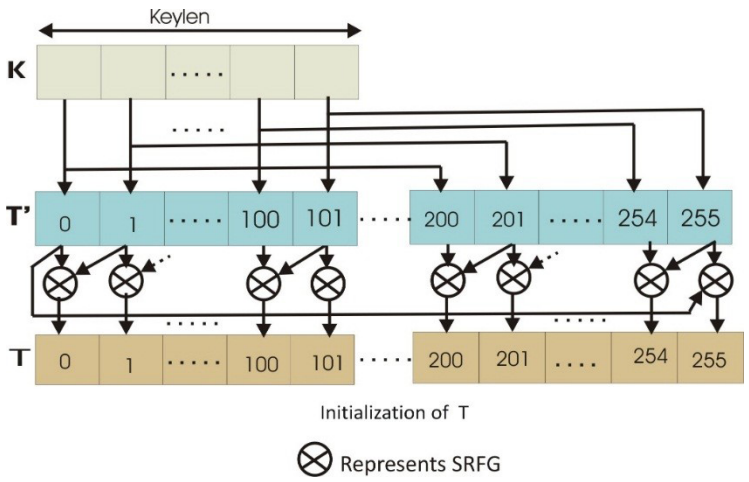

FIGURE 2. The proposed modification in RC4 scheme.

derived from a class of table-shuffling ciphers, and where the table entries are permuted by next-state function.

In [13], the authors present a generalizes framework for differential cryptanalysis on RC4. The differences in the key, or in the keystream patterns, are then used to analyse the bit vectors of the internal state of the cipher and in retrieving that bit sequence. In [14], the authors analyse the permutation operations which are considered to be non-linear. The theoretical analysis of the work shows that permutation bytes in any stage in key scheduling algorithm of RC4 are biased, and that this bias reveals the secret key eventually. The colliding key problem in RC4 has been described in [15]. These keys disrupt the pseudorandomness and are able to calculate the same initial state, and which eventually outputs the same pseudorandom byte stream. A new state transition sequence of the key scheduling algorithm is shown which is used with a related key pair of an arbitrary fixed length, thus leading to key collision problems. Another key collision work on RC4 has been researched in [16], and attack on RC4 (n,m) has been shown in [17]. The authors show two attacks: one is based on non-randomness of internal state, and which allows it to be distinguishes from a truly random cipher. Another attack is depending upon low diffusion of bits in the key scheduling of RSA and PRGA algorithms, and recovers all the bytes of the secret key. Empirical correlations among the keystream bytes and the secret key have been studied thoroughly in [18], and which show that the non-randomness behaviour of RC4 works as a bias and exhibits such relations which are used for cryptanalysis attacks.

A number of improvements to RC4 have been suggested by different researchers. An improved RC4 has been shown in [19] and which uses a new pseudorandom bit generator with two secret keys and three pointers. The key size condition for robustness in RC4 has been analysed in the paper [20], and a analysis of modified RC4 following the same use of two keys has been shown in [21]. Three enhanced variants of RC4 have been proposed in [22], and where the authors concentrate on the pseudorandom number generation, rather than the key scheduling. A hybrid chaotic-based approach has been used in [23] for improving the strength of RC4. In [24], the initial state factorial is used to solve the correlation issue of RC4, and an additional state table is used to remove the correlation between publicly known outputs of the internal state. The same length as that of the state has been used for this table to contain the factorial of initial state elements.

The analysis of the previous work signifies that RC4 suffers from cryptanalysis attacks due mainly to its keystream biasness. Moreover, the collision of keys and the distinguisher generation also create major drawback in this cipher. To mitigate these, we propose the usage of SRFG [6] so that randomness can be strictly provided in key scheduling and no keystream byte can be backtracked to the secret key.

III. MODIFIED RC4 (MRC4) AND ITS FEATURES

The analysis of the literature review exhibits the fact the RC4 faces the cryptanalysis problems due to its key scheduling algorithm. Thus we have modified the first stage of RC4 key scheduling. and where we added SRFG to each byte transferred from K to T. For this, SRFG will get two inputs from two consecutive bytes of K. The bytes in K is considered to be iterative until the completion of bytes of T. For the last byte of T to be generated, the last byte of K (after expanding) and the first byte K is considered as inputs of SRFG. At this time, we have stored the keystream accordingly to bytes for decryption process. The modification has been summarized in Algorithm 1. All the others modules are kept the same and therefore only the modified process is shown in Figure 2.

We have experimented the key scheduling algorithm (KSA) of RC4 with our previously developed SRFG [6]. It removes the biasness from the keystream bytes. This bias generates cryptanalysis attacks deducing linear or differential relations among the states or keystream bytes and backtracks the secret key. We have explained the important features of the outputs of KSA in MRC4 such as: balancedness; non-linearity; resiliency; propagation criterion; and

Algorithm 1

```
procedure Algorithm-1
    for i = 0 to 255 do
    if (keylen == 256) then
        for j = 0 to 255
        T'[j] = k[i]
        end loop
    else
        for j = 0 to keylen
        recursively copy k[j] to T'[i]
        end loop
    end if
    end loop
    for i = 0 to 255 do
    T[i] = T'[i] ⊗ T'[i + 1] where ⊗ represents SRFG
    end loop
    T[255] = T'[255] ⊗ T'[0]
end procedure
```

immunity by applying the SRFG. Each byte b_i in the initial state space T' is comprised of 8 bits and is considered as 8-bit byte vector b in our experimentation.

Consider $\mathcal{F}_2$ as the set of all the functions on *two* variables input to the SRFG stage in generation of T where all the functions mapping from F_2^2 into F_2 providing $F_2^2 = \{(b_1, b_2) | \ b_i \in F_2\}$. F_2 is considered as the finite field of two elements {0,1} and $\bigoplus$ is any operation on the field F_2.

Any combined function $f_c \in \mathcal{F}_2$ of *five* terms (in this proposed work, we have taken the expression length as 5) is expressed as a polynomial and given as:

$$f_c(b_1, b_2) = \oplus\lambda_u \left(\prod_{i=1}^{2} rand(b_i)^{u_i}\right)^5, \quad \lambda_u \in F_2 \mathrm{u} \in F_2^2 \tag{1}$$

with,

$$\lambda_u = \oplus f_c(b), \quad b \preccurlyeq u, \ \forall\, b_i = \{b_{i_1}, \ b_{i_2}, \ \ldots., \ b_{i_8}\} \tag{2}$$

where,

$$(b_{i_1}, b_{i_2}, \ldots, \ b_{i_8}) \preccurlyeq (u_1, u_2, \ldots, \ u_8) \ \textit{iff} \quad \forall i, j \ w_{i_j} \leq u_i \ \textit{and} \ j = 1, 2, .., 8 \tag{3}$$

The output of f_c depends on the weight of its input variables. As a result, f_c corresponds to a function $\mathfrak{h}_c$: {0, , 1, . . . ,8} → F_2 such that $\forall x \in F_2^2$, $f_c(x) = \mathfrak{h}_c(wt(x))$. The sequence $\mathfrak{h}_c(f_c) = (\mathfrak{h}_c(0), \mathfrak{h}_c(1) \ldots, \mathfrak{h}_c(7))$ is also an 8-bit byte vector and considered as simplified value vector of f_c. The established relation between simplified value vector and Arithmetic Normal Form (ANF), can be shown as:

$$f_c(b_1, b_2) = \oplus\lambda_f(j) \oplus \left(\prod_{i=1}^{2} rand(b_i)^{u_i}\right)^5 = \oplus\, \lambda_f(j)\ \mathcal{X}_{j,N} \tag{4}$$

where, $\lambda_f(j)$, u $\in F_2^2$ *and* $L \in \mathbb{Z}$, $j = \{1, 2\}$. $\mathcal{X}_{j,N}$ is the two variable elementary polynomial of degree j. The coefficients of ANF of f_c is represented by $8 - bit$ vector, $\lambda(f_c) = \{\lambda_f(0), \lambda_f(1), \ldots., \lambda_f(8)\}$. The functional and mathematical properties of KSA in MRC4 have been shown in following subsections.

A. BALANCEDNESS

The balanced property in the proposed KSA in MRC4 exists if its simplified value vector $\mathfrak{h}_c$ satisfies the following condition:

$$\forall\, i = \{1, 2\} \quad \mathfrak{h}_c(i) = \mathfrak{h}_c(2 - i) \boxplus 1 \quad \textit{where} \ \boxplus \textit{is sum over} \ F_2 \tag{5}$$

Inheriting the trivial balancedness feature from SRFG in KSA f_c verifies the condition $D_1 f_c = 1$ which actually does not exist for even values of n (here $n = 8$ for bytes). For any byte vector b, $wt(b) = n/2$ (*where*, $wt(b)$ is the weight of byte vector defined as number of 1s in it), we can calculate the $D_1 f_c$ as:

$$D_1 f_c = f_c(b) \boxplus f_c(b + 1) = \mathfrak{h}_c\left(\frac{n}{2}\right) \boxplus \mathfrak{h}_c\left(\frac{n}{2}\right) = 0 \tag{6}$$

As each byte is balanced, the overall bytes in initial states are also balanced. Therefore, this balancedness property helps to prevent attacks depending upon the weight or Hamming distances [25].

B. NONLINEARITY

The nonlinearity is calculated by the Hamming distance between two affine transformations. For example, two-byte vectors are: b_i and b_j of 8 bits each:

$$\mathcal{N}\ell\left(b_{i_k} \ b_{j_k}\right) = \sum_{k=1}^{n} b_{i_k} \neq b_{j_k}, \quad where \ n = 8 \tag{7}$$

This property of non-linearity removes the probability of bias and reduces the chances of deducing relations with all 0s or all 1s inputs. Each byte follows this non-linearity to generate the initial state of T.

C. RESILIENCY

The proposed KSA in MRC4 is *m-resilient* if the output remains balanced when any m input bits are fixed and remaining *(8-m)* bits are altered in next sequence. The function is more resilient if m is higher. The property of resiliency is related to the weights of the restrictions of the f_c to some subspaces.

$\forall\, f_c \in \mathcal{F}_2$ and any affine subspace $\mathcal{S} \subset F_2^2$, the restriction of f_c to $\mathcal{S}$ is the function given as:

$$f_{\mathcal{S}} : \ \mathcal{S} \to F_2 \tag{8}$$

The subspace $\mathcal{S}$ is spanned by k canonical basis vectors and its supplementary subspace is $\overline{\mathcal{S}}$. The restrictions of f_c to $\mathcal{S}$ and to all its cosets are given by $b' + \mathcal{S}$ where, $b' \in \overline{\mathcal{S}}$. Being f_c symmetric and balanced, $\mathcal{S}$ is represented

as: $\mathcal{S} = \langle s_1, s_2, \ldots, s_k \rangle$ and $f_{a+\mathcal{S}}$ becomes symmetric and balanced too. Moreover, for all $s \in \mathcal{S}$, we can write the following:

$$f_{b'+\mathcal{S}}(s) = f\left(b' + s\right) = \mathfrak{h}_c\left(wt\left(b'\right) + wt\left(s\right)\right) \tag{9}$$

The simplified value vector and the simplified ANF vector of $f_{a+\mathcal{S}}$ can be deduced from f_c as given below.

$$\mathfrak{h}_{cf_{b'+\mathcal{S}}}(i) = \mathfrak{h}_c\left(i + wt\left(b'\right)\right), \quad \forall i,\ 0 \le i \le k \tag{10}$$

$$\lambda_{f_{b'+\mathcal{S}}}(i) = \oplus\, \lambda_f\ (i+j), \quad \forall i, \quad 0 \le i \le k \ \text{ and } \ j \preccurlyeq wt(b') \tag{11}$$

Proposition 1: For the proposed KSA in MRC4 with m-resiliency factor, two consecutive keystream bytes with 8 bits each have maximum non-linearity of 5. In such cases, the difference between simplified value vectors of the two consecutive bytes of T equals to the sum of the non-linearity of two consecutive bytes b_i, b_j, in T.

$$\max[\mathcal{NL}\ (b_i\ b_{i+1})] \to 5 \tag{12}$$

$$\left[\mathfrak{h}_{c\mathrm{KSA}}\ (T_k) - \mathfrak{h}_{c\mathrm{KSA}}\ (T_{k+1})\right] = \sum_{i,j=0}^{255} \mathcal{NL}\left(b_i,\ b_j\right) \tag{13}$$

D. PROPAGATION CRITERION

Following the basic properties of SRFG as shown in our previous work [6], the proposed KSA satisfies the propagation criterion of degree k and order m as the outputs of KSA are represented by affine functions keeping m input bits constant and satisfies the propagation criterion of degree k. Considering each keystream byte for experimentation.

Let $f_c \in \mathcal{F}_2$ and let $b_i, b_j \in F_2^2,\ \forall\, i, j = 1, 2, .., 256$, such that $wt\,(b_i) = wt\left(b_j\right) = \frac{n}{2}$. Then, $D_{b_i}f_c$ and $D_{b_j}f_c$ are linearly equivalent. This signifies that if we change the input variables with a linear permutation μ of F_2^2, such that $D_{b_i}f_c = D_{b_j}f_c \circ \mu$, where $\circ$ is composite function. The permutation μ exists on the variable in a way so that $b_j = \mu(b_i)$. Since, f_c used in KSA is symmetric and balanced, we can have,

$$D_{b_j}f_c\ (\mu\,(\mathrm{a})) = D_{b_i}f_c\ (a), \quad \textit{where } a \in \overline{\mathcal{S}} \tag{14}$$

Let k be an integer, $1 \le k \le n-1$, $z \in \overline{b_i} = \left\langle b_{i_1}, b_{i_2}, \ldots, b_{i_{n-k}} \right\rangle$ and $\varepsilon_k = b_{n-k+1} + \cdots + w_n$. Then for any $z = a + b_j$ with $a \in \overline{\mathcal{S}}$, then we can have the following.

$$wt\,(z) = wt\,(a) + wt\left(b_j\right) \tag{15}$$

$$wt\,(z+\varepsilon_k) = wt\,(a) + wt\left(b_j+\varepsilon_k\right) = wt\,(a) + k - wt(b_j) \tag{16}$$

Thus, $\forall\, a \in B,\ B = \{b_0, b_i \ldots,\ b_{255}\}$

$$\begin{aligned} D_{\varepsilon_k}f_c\ (a+y) &= f_c\,(a+b)\ \boxplus f_c\left(a+\varepsilon_k+b_j\right) \\ &= wt\,(a) + wt\left(b_j+\ \varepsilon_k\right) = wt\,(a) + k - wt\left(b_j\right) \\ &= \mathfrak{h}_c\left(wt\,(a) + w\left(b_j\right)\right) \boxplus \left(wt\,(a) + k - w\left(b_j\right)\right) \end{aligned} \tag{17}$$

Equation 17 signifies that $\mathfrak{h}_c$ follows the symmetric property. This means that partial derivatives of our proposed KSA outputs are also propagated with the propagation features.

E. IMMUNITY

Correlation immunity and algebraic immunity are examined for the proposed KSA in MRC4. For correlation immunity, considering each of the two input bytes b_i as a 8-bit binary vector the outputs are correlation immune if:

$$Prob\,(f_c = b_i) = \frac{1}{2}\ 1 \le i \le 8 \tag{18}$$

The probability distribution must be equal for all the bits and therefore, the output byte $b_o \in \mathrm{T}$ having the following property.

$$|\min[M_0\ \left(b_o, (\ b_o)^r\right) - M_1\ \left(b_o, (\ b_o)^r\right)]| = \min[m] \mapsto 0 \tag{19}$$

where, $[M_0\ (b_o, (\ b_o)^r)]$ is the matching of output bytes from KSA and its reverse with respect to value 0 and $[M_1\ (b_o, (\ b_o)^r)]$ is the matching of output words from KSA and its reverse with respect to value 1.

Following the above property, an interesting feature of our proposed key expansion module has been identified and the proposition has been given as:

Proposition 2: In MRC4, if $[M_0\ (b_o,\ (\ b_o)^r)] = m_0$ *and* $[M_1\ (w_o,\ (\ w_o)^r)] = m_1$ then the maximum non-linearity between two successive T_k will be the sum of all m_0 and m_1 for all the b_o in T.

$$max\,[\mathcal{NL}\ (T_i\, T_{i+1}\ )] = \sum_{1}^{256} m_0 + m_1, \quad \textit{for all } b_o \text{ in T}$$

Algebraic immunity is related with the annihilator of a function. To evaluate this property for the proposed KSA we have considered the following. Given, $f_c \in \mathcal{F}_2$, any function of the set $A\,(f_c) = \{\, g \in \mathcal{F}_2 \mid gf = 0 \,\}$ is defined as the annihilator of the function f_c . The algebraic immunity of f_c is denoted by $AI\,(f_c)$ is the minimum degree of all nonzero annihilators of f_c or $f_c + 1$. The value of $AI\,(f_c)$ is given as:

$$AI\,(f_c) = \min\left[\deg\,(g)\right] | \, g \neq 0, \quad g \in A\,(f_c) \cup A\,(f_c+1) \tag{20}$$

As we have used SRFG to generate the output bytes, the minimum degree is always $\frac{n}{2}$. Therefore, the algebraic immunity of the outputs from it is always $\frac{n}{2}$ which is always optimal.

We can rewrite the equation 20 for two consecutive state of T as:

$$\begin{aligned} &AI\,(f_c) = \min[\deg\,(T_i)] | \, f_c \to T_i g \neq 0, \\ &A\,(f_c)\ \cup\ A\,(f_c+1)\ \neq 0 \\ &\vdash AI\,(f_c) - AI\,(f_{c+1}) = 0 \end{aligned} \tag{21}$$

IV. SECURITY ANALYSIS

The analysis of the previous works on RC4 in Section II shows that the cipher suffers from the problem of biased distinguisher and which eventually deduces the relational statistics for identifying the secret key. The biasedness is generated with all the binary zero or all the binary one values

of bytes of inputs. Moreover, related key attacks are also responsible for linear and differential cryptanalysis on RC4. Therefore, in the following subsections we have analysed the security features of MRC4 in the above said perspective.

A. RELATED KEY ATTACK ANALYSIS

We focus on how the S-box initialization responds to a single-byte difference in its input. Assume any random key $k'(i) = k(i)$ where, $k(i)$ is the original key except when $i = t$ where $k(t) \neq k'(t)$. In this case, $j = (j + S(i) + k(i)) mod 256$ will result in different values of j. If t is close to zero, the resulting S-boxes will be completely different. If t is close to 255, however, the S-boxes will be substantially similar because the first *t-1* iterations through the initialization loop have performed exactly the same work. We have used a twiddled key $k'(t) = k(t)+\mu$ and therefore $k'(t+1) = k(t+1)-\mu$, the value of j at i^{th} iteration will be $j'_i = j_i + \mu$ and j'_{i+1} will be same as j_i. This *k'* and *k* are called related keys. With this key-setup, in the initialization phase, the RC4 output for the original key and the related key will proceed in lock step for $b'_i = b_i$ still $s'_{i-1} \neq s_{i-1}$. At this point we define it as derailment of RC4 systems and identical bytes produced by the two keys is defined as the length of the derailment.

In our experiment, for two randomly chosen keys - in the initialization phase of the KSA - the probability of two bytes to be similar is given as:

$$P(b'_i = b_i) = \sum_{i=0}^{255} \bigoplus \prod_{4}^{5} P(b'_i) \times P(b_i) = 256 \times \frac{1}{3125} \times \frac{1}{256} \times \frac{1}{256} = 0.00000125 \quad (22a)$$

To generalize the above equation for any x bytes of key, the above equation can be re-written as:

$$P(b'_i = b_i) = \sum_{i=0}^{i=x-1} \bigoplus \prod_{4}^{5} P(b'_i) \times P(b_i) \quad (22b)$$

The probability of the derailment length of d is calculated as:

$$P(d) = P(b'_i = b_i)^d \rightarrow 0 \ \textit{due to its least value} \quad (23)$$

The derailment of RC4 is related to the *distinguishing attack*. A distinguisher is an algorithm which distinguishes a stream of bytes from a perfectly random byte stream. It verifies whether a stream of bytes that has been chosen is according to the uniform distribution. Cryptanalysts try to distinguish between a string generated by an insecure pseudorandom byte generator and one from a perfectly random source. There are different types of distinguishers used by the adversaries as: regular; prefix; and hybrid. In all the cases, the main objective is to identify a two-byte sequence to be same, in order to observe a higher derailment length. But, in our experimentation, we have found that the probability of two bytes getting similar is too small and therefore the distinguishing attack cannot be inferred.

B. LINEAR CRYPTANALYSIS

We have followed the bit-advantage concept for analysis of linear attacks. If an attack is executed on an n-bit key and recovers the correct value of the key ranked among the top m out of 2^n possible candidates, the attack obtains an $(n - log(m))$ -bit advantage over exhaustive search. In such attacks, linear equations are made from the known plaintexts functioning with the ciphertext to get the key bytes. Linear cryptanalysis is common in RC4 as the key bytes are directly XORed with plaintext bytes and key bytes are achievable. But, in MRC4 we have processed KSA with SRFG so that for any two random bytes of keystream, the total key is not going to be deduced. Therefore, we have analysed the linear relation approximation on two related keys rather than on known plaintexts as below.

Following Theorem 2 in the research work [26], if P_s is the probability that a linear attack on an n-bit keystream byte (8 bits here) - with a linear approximation of probability p and with N known related keystream bytes - delivers an m-bit or higher advantage. Assuming that the linear approximation's probability to hold is independent for each key tried and is equal to $\frac{1}{2}$ for all wrong keys, we can have:

$$P_s = p^N (n - m) \left(1 - 2^{m-1}\right) \quad (24)$$

Now, to calculate the linear approximation of probability p of the attack we have evaluated that whether any two wrong keys k_1 and k_2 are deriving two similar bytes or not. The probability of such similarity has been calculated in Equation 25. This shows that the success probability P_s is also reduced and approximates to 0 if $m \rightarrow n$.

$$P_s = p^N (n - m) \left(1 - 2^{m-1}\right) = (\approx 0)^N (n - n) \left(1 - 2^{n-1}\right) = 0 \quad (25)$$

Therefore the proposition can be written as:

Proposition 3: In MRC4, if $P(b'_i = b_i) \rightarrow 0$ *and n-bit random keystream byte attempts to generate an n-bit advantage, the success probability* $P_s = 0$.

C. DIFFERENTIAL CRYPTANALYSIS

We concentrate on keys of 256 bits as these keys are very common in wireless implementations. The differential cryptanalysis deduces the output streams that are expected to be the same in the first few bytes even though input has been changed. Generally, this cryptanalysis is a type of chosen plaintext. Therefore, we need to check the probability of getting any two similar cipher text bytes for two chosen plaintext bytes.

Let pt'*and* pt'' are two chosen plaintext of l bytes with a known difference MRC4 and given as:

$$pt' - pt'' = \mathcal{N}\ell(\text{pt}', \ \text{pt}'') \quad (26)$$

Due to the convenience of calculation, we assume that the non-linearity is uniformly distributed among all the bytes of

the chosen plaintexts and therefore, from Equation 27 we can write the following:

$$\mathcal{N}\ell\,(\,\mathrm{pt}'_i, \mathrm{pt}''_i) = \frac{\mathcal{N}\ell(\,\mathrm{pt}', \mathrm{pt}'')}{l} = \Delta_i, \quad \text{for } i = 0, 1, 2, .., l-1$$

Following the Equation 22b, the probability of any two ciphertext bytes to be similar can be calculated as:

$$P(ct'_i = ct''_i) = \frac{1}{\frac{l!}{2!\times(l-2)!}} \times P(b'_i = b_i) \times \frac{1}{\Delta_i} \quad (27a)$$

Assuming a key size of 256 bits (32 bytes) and plaintext size of 64 bytes, we can write from the above equation is:

$$\begin{aligned} &P(ct'_i = ct''_i) \\ &= \frac{1}{\frac{l!}{2!\times(l-2)!}} \times P(b'_i = b_i) \times \frac{1}{\Delta_i} \\ &= \frac{1}{2016} \times 32 \times \frac{1}{3125} \times \frac{1}{32} \times \frac{1}{32} \times \frac{1}{\Delta_i} \\ &= 5 \times 10^{-9} \times \frac{1}{\Delta_i} \end{aligned} \quad (27b)$$

The above equation shows that using the two chosen plaintext, the probability of two random similar cipher bytes is impractical. Therefore, differential analysis is not possible on our MRC4. From, this analysis, we can infer the following proposition.

Proposition 4: In MRC4, the probability of the two similar ciphertext bytes is inversely proportional to the number of plaintext bytes and inversely proportional to non-linearity and given as:

$$P(ct'_i = ct''_i) \propto \frac{1}{\Delta_i} \quad (28a)$$

$$P(ct'_i = ct''_i) \propto \frac{1}{l} \quad (28b)$$

From the above, we can write:

$$\begin{aligned} &P(ct'_i = ct''_i) \propto \frac{1}{\Delta_i \times l} \vdash P(ct'_i = ct''_i) \\ &= k \times \frac{1}{\Delta_i \times l} \text{ where } k \text{ is constant and given as } k \\ &= (b'_i = b_i) = 0.125 \times 10^{-4} \end{aligned} \quad (29)$$

S-box information maps the non-zero inputs to any non-zero outputs in the array. For the RC4 with 256 bits, 256 states are defined. Using the SRFG the randomization for temporary state generation T' also obtains the randomized bits for every iteration with $P(b'_i = b_i)$ measured very small. We have compared some of the RC4 versions for the process and measured the active S-boxes in multiple iterations shown in Table 1.

D. NON-RELATED KEY ATTACKS

In this attack environment, the same data is encrypted several times using the same key, but using a different Initial Value (IV). This is referred to as the standard (non-related-key) model, where the IV value is presumed to be under control of the attacker.

TABLE 1. S-box information.

Algorithm name	$P(b'_i = b_i)$	Active S-boxes
Original RC4	0.564 after 50% of the 256 states of repetition	128
Werrasinghe [21]	0.33 after 60% of the 256 states of repetition	144
RC4-M3 [22]	0.27 after 70% of the 256 states of repetition	128
RC4-A [11]	0.15	190
VMPC [32]	0.23	200
RC4+ [33]	0.31	128
Proposed MRC4	≈ 0	256
Jindal et. al. [34]	0.37	141

A plaintext pt is encrypted several times using different initial values $V_1, V_2, .., V_n$ which are biased and under control of attacker. Assuming the length of the initial values V_i are of b bits, and probability of derailment rail identification is approximately zero, the probability of two ciphertexts to be similar can be derived as an extension of linear cryptanalysis and relate key analysis. It is calculated as:

$$P(ct' = ct'') = P(d) \times P(b'_i = b_i) \to 0 \quad (30)$$

As $P(d) \to 0$ and $P(b'_i = b_i)$ is small enough for large number of bits sequence, the overall probability of the two ciphertexts to be same also near to zero which signifies that the non-related key attacks are also resistible by the presented MRC4.

V. PERFORMANCE EVALUATION

We have implemented the proposed MRC4 in software with hardware specification as: CPU: 2.6 GHz, i3 6th Gen with 4 GB RAM. MRC4 is developed with Python using cryptographic libraries. The output results are analyzed using R programming and NIST statistical suite. We have compared our experimental results with the original RC4, four extended versions of RC4 [11], [22], [32], [33] and two recent improvements of RC4 [21], [34]. We have stored the key bytes generated from the keystream generator and we shall consider this as a feature work to deal with key storage process. Note that, only the keystream generation process is only modified in MRC4; however, we have measured the results with the overall process of encryption and decryption. We have varied the key sizes from 64 bits to 2,048 bits in all the performance metrics.

The first comparison is done on the basis features extracted in Section III: non-linearity, balancedness, resiliency, propagation criterion, correlation immunity and algebraic immunity. We have measured the order of the features stated above and compared accordingly as in Table 2. The comparison results in Table 2 signifies that our proposed modification of keystream generation is efficient in terms of the above said features. We can see from the table that the original RC4 method was lagging behind in acquiring the important features of cryptographic functions and therefore,

TABLE 2. Comparison of features.

	Order of Non-linearity	Order of Balanced-ness	Order of Resiliency	Order of Propagation criterion	Order of Correlation immunity	Order of Algebraic immunity
Original RC4	$\frac{n}{10}$	0	0	0	$\frac{n}{10}$	0
Weerasinghe [21]	$\frac{n}{6}$	0	0	0	$\frac{n}{4}$	0
RC4-M3 [22]	$\frac{n}{4}$	0	$log_2 n$	$\frac{log_2 n}{n}$	n	$\frac{n}{8}$
RC4-A [11]	$\frac{n}{6}$	0	0	$log_2 \frac{n}{3}$	$\frac{n}{8}$	$\frac{n}{8}$
VMPC [32]	$\frac{n}{6}$	0	0	$log_2 \frac{n}{2}$	$\frac{n}{10}$	$\frac{n}{8}$
RC4+ [33]	$\frac{n}{4}$	0	$log_2 n$	$\frac{log_2 n}{n}$	$\frac{n}{8}$	$\frac{n}{4}$
Jindal et. al. [34]	$\frac{n}{8}$	0	$log_2 \frac{n}{2}$	$\frac{log_2 n}{n}$	$\frac{n}{4}$	$\frac{n}{10}$
Proposed MRC4	$\frac{n}{3}$	$\frac{n}{2}$	$\frac{n}{2} - 2$	$log_2 n$	n	$\frac{n}{2}$
n is the value of bits in cipher text considering all the bits, orders are calculated based upon maximum degree						

TABLE 3. Time Consumption of KSA.

Hardware specification for computation: CPU: 2.6 GHz, i3 6th Gen with 4 GB RAM						
Scheme	*Key size*					
	64 bit	*128 bit*	*256 bit*	*512 bit*	*1024 bit*	*2048 bit*
Original RC4	20.32	18.39	17.58	17.23	16.42	16.13
Weerasinghe [21]	25.35	20.15	19.7	18.71	17.45	16.57
RC4-M3 [22]	44.42	43.73	43.09	42.66	40.78	40.32
RC4-A [11]	43.90	41.25	39.87	38.66	37.50	33.87
VMPC [32]	45.88	45.01	43.20	42.87	41.67	41.50
RC4+ [33]	40.90	39.25	37.87	36.67	35.50	33.87
Jindal et. al. [34]	38.82	37.37	36.87	35.50	34.33	33.67
Proposed MRC4	43.47	42.83	41.16	40.27	39.50	39.20

TABLE 4. Comparison of cost of attacks.

	Differential cryptanalysis	Linear cryptanaly-sis	Related key at-tacks
Original RC4	2^{128}	2^{64}	2^{128}
Weerasinghe [21]	2^{128}	2^{128}	2^{128}
RC4-M3 [22]	2^{512}	2^{128}	2^{512}
RC4-A [11]	2^{178}	2^{512}	2^{512}
VMPC [32]	2^{712}	2^{430}	2^{512}
RC4+ [33]	2^{908}	2^{512}	2^{624}
Jindal et. al. [34]	2^{512}	2^{128}	2^{228}
Proposed MRC4	2^{1024}	2^{1024}	2^{3125}

the attacks as considered in the previous work are executable on this cipher. Better results are noticed in improvement proposed in [21] as compared to original RC4, but still it is not enough for the purpose. Furthermore, the improvement shown in [22] and the other versions of RC4 modifications as in [11], [32]–[34] show far better results as compared to the previous two algorithms and possess some cryptographic attributes. However, the use of SRFG in the proposed modification has provided improved features and the optimality of balancedness which is useful for preventing bitsum attacks [25]. The high correlation immunity in the proposed MRC4 is advantageous to prevent correlation attacks [27].

We have compared the computation time for all the algorithms as stated above on the same platform. In this comparison too, we have assumed the time for XORing operation is a constant for both encryption and decryption and therefore not considered in the time consumption calculation. Therefore, Table 3 only compares the time taken (in microseconds) for the Key Scheduling Algorithm (KSA) by varying the key size. The time comparison results show that use of the SRFG in RC4 key scheduling modification is increasing the time consumption in generating the keystream bytes and thus contributing to the trade-off between security and time consumption. The same has also been addressed by the authors in [22]. To support this trade-off and overcome with the security issues, we have also compared the attacks on the algorithms in terms of cost of bits as shown in Table 4. For this comparison, we have considered 50 plaintexts of 128 bytes each with 128-bit (16-byte) keys. Table 4 describes the fact that the cost of the attacks for our proposed MRC4 is much higher than the other algorithms due to the use of randomness with SRFG in KSA. This signifies that MRC4 is better in terms of security.

The time comparison results show that use of the SRFG in RC4 key scheduling modification is increasing the time consumption in generating the keystream bytes and thus contributing to the trade-off between security and time consumption. The same has also been addressed by the authors in [22]. To support this trade-off and overcome with the security issues, we have also compared the attacks on the

algorithms in terms of cost of bits as shown in Table 4. For this comparison, we have considered 50 plaintexts of 128 bytes each with 128-bit (16-byte) keys.

Table 4 describes the fact that the cost of the attacks for our proposed MRC4 is much higher than the other algorithms due to the use of randomness with SRFG in KSA. This signifies that MRC4 is better in terms of security.

Lastly, we have compared two prime evaluation parameters of cryptographic algorithms: confusion and diffusion. According to Shannon's theory [28], confusion property exhibits the statistical relationship of between the ciphertext and key to be more complex whereas, diffusion property yields the relationship between plaintext and ciphertext such that a single bit change in plaintext must change at least half of the cipher text bits. We have introduced two new metrics, Confusion Index (CI) and Diffusion Index (DI) in this point to have a bounded range of the metric and given as:

$$CI = \frac{key\ bits\ \times\ nonlinearity\ in\ two\ ciphertext}{ciphertext\ bits}$$
$$DI = \frac{nonlinearity\ in\ two\ ciphertext}{plaintext\ bits} \tag{31}$$

The maximum value of the confusion index in Equation 31 can be $\propto$ which is not possible in the real-life implementation scenarios as all the cipher text bits cannot be changed for a single bit change in key. If it happens then, it will work as a bias in next state initialization. We have varied the key bits from 64 to 2048 with a fixed plaintext of 128 bytes (1024 bits) and, as an output, we have received 128 bytes (1024bits).

The results are compared with other algorithms as well and shown in Figures 3 and 4 respectively. From Figure 3, we can observe the proposed MRC4 is best in providing confusion as compared to others. Jindal et. al. [34] shows an increasing confusion index but drops down after 1024 bits of keysize. The other algorithms are show significantly low confusion index which is not suitable for stream ciphers. Moreover, an interesting behaviour of MRC4 has been observed here. It shows that up to 128-bit keys, the confusion index is almost static and with the key size of 256 bits to 512 bits it

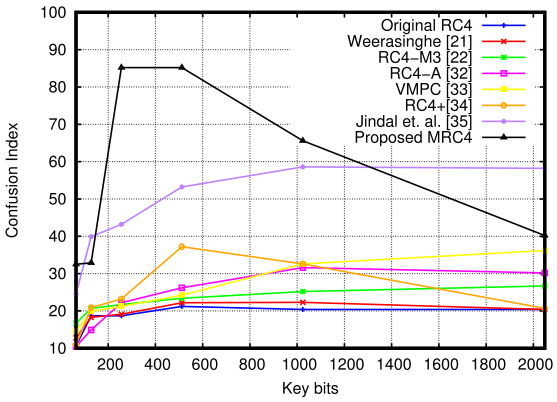

FIGURE 3. Confusion index comparison.

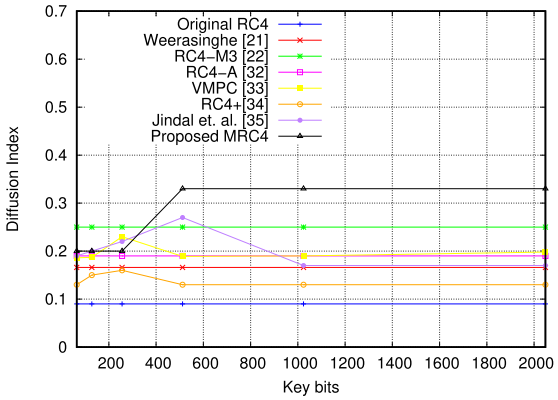

FIGURE 4. Diffusion index comparison.

increases with a high slope and with higher than 512 bits of keys, confusion index decreases rapidly. This fact signifies that for MRC4 using the key sizes in between of 256 to 512 bits is most efficient. Similarly, Figure 4 shows that original RC4 and the algorithm shown in reference [21] are having very low diffusion property. The algorithm in [22] is having a better diffusion index whereas the modification in RC4 shown in [34] provides good diffusion till near about 512 bits and then degrades the performance with increasing keysize. The other algorithms are having average diffusion but less than that of the our proposed MRC4. Like confusion, here also MRC4 follows a constant diffusion index up to the key size of 256 bits but drastically changes with the use of 512 bits of keys and gets constant after with higher bits of keys. Analysing the behaviour of MRC4 in confusion and diffusion, we can say that the use of 512 bits key is considered to be the most efficient use in MRC4 to get high cryptographic features.

VI. CONCLUSION AND FUTURE WORK

RC4 has been a popular stream cipher but has been identified with a number of cryptographic weaknesses. It is, though, well matched to light-weight cryptography methods. This paper outlines a solution related to the randomness in key scheduling algorithm of RC4 and uses SRFG. The experimental results show that MRC4 possess a 60% better confusion property and 50% better diffusion as compared to the original RC4 method. The limitation of this paper is about the time taken by the KSA as compared to the state-of-the-art. Therefore, it is confirmed that RC4 is able to provide good security features at the cost of increased time, and which leads to a trade-off between security and time. Ignoring the fact of the time consumption if we consider only the robustness and security features. MRC4 is efficient in all respects of cryptographic attributes and security evaluation. Furthermore, being a symmetric keystream cipher, RC4 uses the single key for both encryption and decryption. In this paper, the keystream bytes are stored separately for decryption in later stage as required. In future work, it should be possible to analysing the trade-off between storing of random keys, the security and the related space complexity.

AUTHOR CONTRIBUTIONS

R. Saha and G. Geetha conceived the idea, designed the experiments and analyzed the data; G. Kumar, T.H. Kim and R. Saha performed the experiments and conducted the analysis; R.Saha, G. Kumar and W.J. Buchanan analysed the methods, interpreted the results and drew the conclusions; G. Geetha and W.J. Buchanan proof read the paper. All the authors agree with the above contribution details.

CONFLICTS OF INTEREST

All authors declare no conflict of interest.

REFERENCES

[1] W. Stallings, *Network Security and Cryptography*, 5th ed. London, U.K.: Pearson, 2006.
[2] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, "Public key authentication and key agreement in IoT devices with minimal airtime consumption," *IEEE Embedded Syst. Lett.*, vol. 9, no. 1, pp. 1–4, Mar. 2017, doi: 10.1109/LES.2016.2630729.
[3] S. Raza, L. Seitz, D. Sitenkov, and G. Selander, "S3K: Scalable security with symmetric keys—DTLS key establishment for the Internet of Things," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1270–1280, Jul. 2016, doi: 10.1109/TASE.2015.2511301.
[4] F. Chen and Y. Luo, *Industrial IoT Technologies and Applications: Second EAI International Conference, Industrial IoT 2017, Wuhu, China, March 25–26, 2017, Proceedings*. Springer, 2017.
[5] T. W. Cusick and P. Stănică, *Cryptographic Boolean Functions and Applications*. New York, NY, USA: Academic, 2009, doi: 10.1016/B978-0-12-374890-4.00012-4.
[6] R. Saha and G. Geetha, "Symmetric random function generator (SRFG): A novel cryptographic primitive for designing fast and robust algorithms," *Chaos, Solitons Fractals*, vol. 104, pp. 371–377, Nov. 2017.
[7] P. Jindal and B. Singh, "RC4 encryption-a literature survey," *Procedia Comput. Sci.*, vol. 46, pp. 697–705, Jan. 2015.
[8] A. Grosul and D. S. Wallach, "A related-key cryptanalysis of RC4," Dept. Comput. Sci., Rice Univ., Houston, TX, USA, Tech. Rep. TR-00-358, 2000.
[9] S. R. Fluhrer and D. A. McGrew, "Statistical analysis of the alleged RC4 keystream generator," in *Fast Software Encryption*. Berlin, Germany: Springer, 2000, pp. 19–30.
[10] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, 2001, pp. 1–24.
[11] S. Paul and B. Preneel, "A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 3017. New York, NY, USA: Springer-Verlag, 2004, pp. 245–259.
[12] V. Tomaševic, S. Bojanić, and O. Nieto-Taladriz, "Finding an internal state of RC4 stream cipher," *Inf. Sci.*, vol. 177, no. 7, pp. 1715–1727, 2007, doi: 10.1016/j.ins.2006.10.010.
[13] E Biham and O Dunkelman, "Differential cryptanalysis of stream ciphers," IACR, New Delhi, Indian, Tech. Rep. CS-2007-10, 2007.
[14] G. Paul and S. Maitra, "Permutation after RC4 key scheduling reveals the secret key," in *Proc. Int. Workshop Sel. Areas Cryptogr. (SAC)*, 2007, pp. 360–377.
[15] M. Matsui, "Key collisions of the RC4 stream cipher," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 5665, O. Dunkelman, Ed. Berlin, Germany: Springer, 2009.
[16] J. Chen and A. Miyaji, "Novel strategies for searching RC4 key collisions," *Comput. Math. Appl.*, vol. 66, pp. 81–90, Aug. 2013.
[17] M. A. Orumiehchiha, J. Pieprzyk, E. Shakour, and R. Steinfeld, "Cryptanalysis of RC4(n, m) stream cipher," in *Proc. 6th Int. Conf. Secur. Inf. Netw.*, New York, NY, USA, 2013, pp. 165–172, doi: 10.1145/2523514.2523615.
[18] S. Sarkar, "Proving empirical key-correlations in RC4," *Inf. Process. Lett.*, vol. 114, pp. 234–238, May 2014.
[19] J. Xie and X. Pan, "An improved RC4 stream cipher," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling (ICCASM)*, Taiyuan, China, Oct. 2010, pp. V7-156–V7-159, doi: 10.1109/ICCASM.2010.5620800.
[20] B. H. Kamble and B. B. Meshram, "Robustness of RC4 against differential attack," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 14, pp. 2278–1323, Jun. 2012.
[21] T. D. B. Weerasinghe, "An effective RC4 stream cipher," in *Proc. IEEE 8th Int. Conf. Ind. Inf. Syst.*, Peradeniya, Sri Lanka, Dec. 2013, pp. 69–74.
[22] P. Jindal and B. Singh, "Optimization of the security-performance tradeoff in RC4 encryption algorithm," *Wireless Pers. Commun.*, vol. 92, no. 3, pp. 1221–1250, 2017, doi: 10.1007/s11277-016-3603-3.
[23] A. T. Sadiq, A. K. Farhan, and S. A. Hassan, "A proposal to improve RC4 algorithm based on hybrid chaotic maps," *J. Adv. Comput. Sci. Technol. Res.*, vol. 6, no. 4, pp. 74–81, Dec. 2016.
[24] S. M. Searan and A. M. Sagheer, "Modification of RC4 algorithm by using two state tables and initial state factorial," *I. J. Comput. Netw. Inf. Secur.*, vol. 8, no. 12, pp. 1–8, 2016.
[25] B. Amandeep and G. Geetha, "Analyzing the applicability of bitsum algorithm on LSB steganography technique," in *Security in Computing and Communications*, vol. 625. Singapore: Springer, 2016.
[26] A. A. Selçuk, "On probability of success in linear and differential cryptanalysis," *J. Cryptol.*, vol. 21, pp. 131–147, Jan. 2008, doi: 10.1007/s00145-007-9013-7.
[27] A. Canteaut and M. Naya-Plasencia, "Correlation attacks on combination generators," *Cryptogr. Commun.*, vol. 4, nos. 3–4, pp. 147–171, 2012.
[28] E. C. Shannon, "A mathematical theory of cryptography," *Bell Syst. Tech.*, Sep. 1945. [Online]. Available: https://www.iacr.org/museum/shannon/shannon45.pdf
[29] A. Biryukov and L. Perrin, "State of the art in lightweight symmetric cryptography," *IACR Cryptol. ePrint Archive*, vol. 2017, p. 511, 2017.
[30] W. J. Buchanan, S. Li, and R. Asif, "Lightweight cryptography methods," *J. Cyber Secur. Technol.*, vol. 1, nos. 3–4, pp. 187–201, 2017.
[31] W. J. Buchanan, *Wireless Cryptography and Stream Ciphers*. Denmark, U.K.: River, 2017.
[32] B. Zoltak, "VMPC one-way function and stream cipher," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 3017. New York, NY, USA: Springer-Verlag, 2004, pp. 210–225.
[33] S. Maitra and G. Paul, "Analysis of RC4 and proposal of additional layers for better security margin," in *Progress in Cryptology—INDOCRYPT* (Lecture Notes in Computer Science), vol. 5365. New York, NY, USA: Springer-Verlag, 2008, pp. 27–39.
[34] P. Jindal and B. Singh, "Performance analysis of modified RC4 encryption algorithm," in *Proc. Int. Conf. Recent Adv. Innov. Eng. (ICRAIE)*, Jaipur, India, May 2014, pp. 1–5.

RAHUL SAHA received the B.Tech. degree in computer science engineering from the Academy of Technology, West Bengal, and the M.Tech. and Ph.D. degrees from Lovely Professional University, Punjab, India, with the area of specialization in cryptography, position, and location computation in wireless sensor networks. He is currently working as an Associate Professor with the Lovely Professional University. He has many publications in well-renowned international journals and conferences.

G. GEETHA is currently working as a Professor with Lovely Professional University, Punjab, India. Her research interests include security and cryptography, cyber-physical systems, and software engineering.

GULSHAN KUMAR received the B.Tech. degree in computer science engineering from the Amritsar College of Engineering, Amritsar, in 2009, and the M.Tech. and Ph.D. degrees from Lovely Professional University, Punjab, India, with the area of specialization in position and location computation in wireless sensor networks. He is currently working as an Associate Professor with Lovely Professional University. He has many publications in well-renowned international journals and conferences.

TAI-HOON KIM received the B.E. and M.E. degrees from Sungkyunkwan University, South Korea, and the Ph.D. degrees from the University of Bristol, U.K., and the University of Tasmania, Australia. His main research areas are security engineering for IT products, IT systems, development processes, and operational environments.

WILLIAM J. BUCHANAN is a Scottish computer scientist. He currently leads the Centre for Distributed Computing and Security, Edinburgh Napier University, where he is also a Professor with the School of Computing.