Wireless Pers Commun (2017) 92:1221–1250
DOI 10.1007/s11277-016-3603-3

Optimization of the Security-Performance Tradeoff in RC4 Encryption Algorithm

Poonam Jindal[1] · **Brahmjit Singh**[1]

Published online: 17 August 2016

Abstract In this paper, we have investigated different vulnerabilities in RC4 and its enhanced variants to overcome the security attacks. It is established that in spite of several proposals, RC4 is not secure enough and a trade-off is always sought between security and network performance for overall provisioning of the secure communication. The main goal of the work presented in this paper is the optimization of security-performance tradeoff. We have proposed three RC4 variants referred to as RC4-M1, RC4-M2 and RC4-M3. Security of the proposed schemes is analyzed in terms of randomness and computational complexity. All the proposed variants qualify the NIST statistical test suite of randomness satisfactorily. The proposed schemes also offer computational complexity in terms of greater number of operations relative to the existing variants. The strength of the proposed schemes has been analyzed against different cryptanalytic attempts and shown the resistance of proposed schemes against attacks. The security-performance tradeoff has been analyzed in terms of run time, CPU cycles consumed, energy cost, and throughput. Encryption time of the proposed schemes—RC4-M1, RC4-M2 and RC4-M3 is 30.1, 10 and 48.7 % less as compared to RC4+ respectively. The results clearly indicate that the computation load of the proposed variants is significantly reduced as compared to the RC4+, concluding that the proposed schemes are computationally efficient. Our results and their analysis also recognize the suitability of the security algorithms for particular application areas.

Keywords Complexity · RC4 · Randomness · Secret key encryption · Security attacks

✉ Poonam Jindal
poonamjindal81@nitkkr.ac.in; poonamjindal81@yahoo.co.in

Brahmjit Singh
brahmjit.s@gmail.com

[1] Electronics and Communication Engineering Department, National Institute of Technology, Kurukshetra 136119, India

1 Introduction

The ease of implementation, flexibility and mobility provided by Wireless Local Area Networks (WLANs) makes the community rely heavily on wireless communication services in their daily lives. People have started using wireless communication services in accomplishing their important and sensitive tasks. While offering the great convenience, the open nature of wireless medium makes the transmitted data vulnerable to security attacks. To prevent these security attacks many cryptographic primitives including symmetric, asymmetric and without key ciphers, have been developed [1]. These primitives maintains the features of confidentiality, integrity and availability of the transmitted data. Symmetric key ciphers include block and stream ciphers. DES, IDEA, RC5, AES, BLOWFISH, TWOFISH are the various available block ciphers, where a block of data is processed at a time. Whereas in stream ciphers, bit by bit processing is carried out. The different available stream ciphers are RC4, E0 (in Bluetooth), A5/1 and A5/2 (in GSM), SNOW 3G, ZUC (4G), Rabbit, FISH, and HC-256 etc. [2–8].

RC4 is one of the most extensively used stream cipher. In spite of being publicly revealed, the design simplicity of the cipher makes the research community always attracted towards it [9]. The cipher is broadly used in a number of software and web applications including Wireless Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), and Secure Socket Layer (SSL) protocols, Microsoft windows, Apple OCE (*Apple Open Collaboration Environment*), secure SQL (a server for database management) etc. However, the design simplicity, statistical weaknesses and non-random behavior between the key, Cipher text (CT), and Plain text (PT) leaves the cipher open to different security attacks.

To remove the weaknesses and related cryptanalytic attempts, different RC4 variants have been proposed in the literature. But the recent cryptanalytic attempts prove that in spite of a number of efforts in improving the security of RC4, weaknesses still exist in the cipher. It shows that even after the eras of research, the RC4 stream cipher is still an insecure cipher and continues to offer plethora of research problems to the research community.

A work focused on the upgrading of RC4 from its byte oriented design to higher word designs is reported in [10]. The series of improvements were reported in the literature with the names Pypy and TPypy [11, 12]. However all the improvements are changing the basic design of RC4 to great extent and is considered as a big drawback since the time-tested security performance of RC4 is not carrying forward. Arguments can be made on the choice of RC4 instead of stream ciphers listed in e-STREAM project. Working on the e-STREAM finalists [7, 8] is rather more practical than modifying RC4. The e-STREAM finalists are word (32 bit) oriented ciphers and possess more complicated structure and have changed the basic design of RC4. Our goal in the present work is not to contest with these algorithms but to improve the strength of RC4 while keeping its simple design by including few additional operations.

In this paper, three modified designs of RC4 have been presented while keeping the original structure of RC4 as it is, and added some additional operations to increase the security of the cipher as compared to its existing variants. The other major issue that has been addressed in this paper is the security performance tradeoff. Increase in security or the computational complexity of any cipher results in more consumption of resources, which further decrease the network performance. Since a number of modifications have been carried out on improving the strength of RC4 but security performance tradeoff has not

taken much into consideration. We have analyzed the security performance tradeoff and have optimized it to significant extent.

First we have implemented conventional RC4 and its three existing variants. Further we have proposed three new RC4 variants; RC4-M1, RC4-M2, and RC4-M3. The performance analysis of all the existing and modified RC4 variants has been carried out in terms of security and network performance. We have analyzed the security in terms of computational complexity and randomness offered by the cipher. Where, computational complexity is the measure of security and defined as the computational effort required by the intruder to break the cipher. The strength of the proposed schemes against different cryptanalytic attempts has been analyzed theoretically and shown that the proposed schemes are resistant to many attacks that are possible on conventional ciphers. Randomness analysis is performed by performing statistical tests using National Institute of Standards and Technology Statistical Test Suite (NIST STS) on RC4 keystream [13]. It is observed that all the three proposals are more secure than conventional RC4 both in terms of randomness and complexity. All the proposed ciphers are passing the randomness tests of NIST STS, hence prove the statistical security of the cipher which is the root to several attacks. Since no significant attacks have been reported on RC4+ [14], it is considered to be the most secure software RC4 implementation till date but with poor performance in terms of resource utilization. The proposed modifications are developed with the focus of achieving security either equivalent to or more than RC4+ with optimized performance. On the basis of complexity, security offered by RC4 PRGA and RC4 KSA varies in a manner such that RC4-M2 > RC4+ > RC4-M1 > RC4-M3 > RC4 PRGA and RC4-M1 > RC4+ = RC4-M2 = RC4-M3 > RC4 KSA respectively.

Our proposed RC4 variants are more focused on security performance tradeoff optimization. Performance of RC4 is analyzed in terms of execution time, energy cost, CPU cycles and throughput. The obtained numerical values show that the execution time, energy cost, and CPU cycles consumed in the proposed variants are more than conventional RC4 but less than RC4+ and the obtained throughput for the proposed ciphers is less than plain RC4 and more than RC4+. The comparison of proposed schemes with plain RC4 and further plain RC4 and RC4+ clearly reflects the security performance tradeoff. This tradeoff has been taken care of in our proposed schemes. In the proposed schemes parallel outputs bytes are generated that leads to improved performance. Encryption time for the proposed schemes—RC4-M1, RC4-M2 and RC4-M3 is 30.1, 10 and 48.7 % less as compared to RC4+. It depicts that the computation load of the proposed variants has been reduced significantly as compared to the RC4+, hence the proposed schemes are computationally efficient schemes. We have achieved security with performance better than RC4+, without affecting the basic design of RC4.

Further for real time applications, information security while achieving good network performance is always desirable. There are variety of network services which varies in their security and QoS requirements. Our analysis is useful in understanding the applicability of RC4 in real time applications. The comprehensive performance analysis reported in the paper may be used as reference for selecting the RC4 variant for given applications or service required.

The rest of the paper is organized as follows: the brief discussion regarding the choice of RC4 is presented in Sect. 2. Section 3 reviews the existing weaknesses, and the related cryptanalysis of conventional RC4. Various enhancements in RC4 are presented in Sect. 4. Section 5 describes the working of existing RC4 encryption algorithms. Our proposed RC4 variants are discussed in Sect. 6. Security analysis and performance analysis of all the variants is elaborated in Sects. 7 and 8 respectively. Recommendations of the proposed

schemes for various application scenarios are made in Sect. 9. Conclusion and future scope are drawn in Sect. 10.

2 Why RC4?

Working on the e-STREAM finalists-HC-128, SALSA 20/12, Rabbit, SOSEMANUK has been shown more practical approach rather than modifying the RC4 [7, 8]. The choice of RC4 has been made amongst all the modern stream ciphers because of the following properties;

- RC4 is the most widely accepted stream cipher specifically in web and network security.
- It offers commendable simplicity, ease of implementation both in software and hardware, speed and efficiency.
- Can be stated in few lines.
- In spite of the decades of analysis, community is still more and more curious about the strengths and weaknesses of the algorithm.
- It is the only software stream cipher with patented hardware structure.
- In contrast, the e-STREAM finalists are word (32 bit) oriented and possess more complicated structure and resulting modifications in the basic design of RC4.

The above discussion clearly corroborates that there is no other stream cipher with surprising characteristics both in software and hardware implementations in comparison to RC4. However, we have analyzed the performance of HC-128 as presented in Sect. 8.

3 Weaknesses of RC4 Stream Cipher

RC4 designed in 1984 and publicly revealed in 1994, is vulnerable to different security attacks. Since then, weaknesses of the cipher have been exploited for having access on either input state or key. In RC4 algorithm, PRGA generates a random sequence of bytes from the scrambled internal state which itself is a random sequence. An intruder while attacking RC4, always focus on the non-random behavior either in the internal state or in the output keystream. In depth discussion on the various weaknesses of RC4 algorithm, which are the roots to several attacks have been reported in [15]. These weaknesses include weak keys, key collisions, key recovery from state, key recovery from key-stream, state recovery and biased bytes. Cryptanalytic attempts pertaining to weak keys were given by Roos and Wagner in [16–18]. The construction and cryptanalysis of related key pairs (key collision) that produce similar state and in turn the similar output keystream even if two different keys are used has been carried out by authors in [19–23]. The reversible nature of RC4 PRGA was explored for the first time by Paul and Maitra [24] and was motivated by the observation made by Roos [16], that key bytes and the PRGA state bytes are correlated. A better key recovery approaches based on differential equations, equation solving approach, modular equations and bidirectional search are discussed in [25–28]. Another RC4 weakness i.e. to recover key from output keystream, was exploited when used in WEP and WPA. Various attacks on WEP (RC4 is used as an encryption algorithm) reported in the literature are Fluhrer, Mantin and Shamir (FMS) [29], Korek practical [30, 31], Mantin [32], Klein [33], Tews, Weinmann and Pyshkin

(TWP) [34], Vaudenay and Vuagnoux (VV) [35], Tews and Beck (TB) [36], Sepehrdad, Vaudenay and Vuagnoux (SVV) [37–39], and Sepehrdad, Susil, Vaudenay and Vuagnoux (SSVV) attacks [40]. Due to these attacks, WEP was considered to be an insecure protocol and was replaced by WPA. Though the protocol removes several attacks in WEP, but the existence of TB data injection [36], and SVV attacks [39] made the protocol insecure. Further, WPA2, a more secure protocol, was proposed where AES encryption is used. Though WPA2 is considered to be a secure protocol, but is not cost efficient as compared to WEP and WPA (where RC4 was used as a basic module). Further, the state recovery is possible in RC4, in spite of its huge state space i.e. $256! \times 256^2 \approx 2^{1700}$. In Knudsen et al. [41] proposed the first state recovery attack on RC4. Different approaches for analyzing the state recovery attacks are reported in [42–47]. Another possible attacks on RC4 are due to its biased bytes. The knowledge of such bytes is always the goal of an attacker. Several biases related to secret key, state variables, and short term and long term biases related to keystream bytes are elaborated in [48–62].

The available literature presents a number of security vulnerabilities in the RC4. But the robustness and design simplicity of RC4 has made it the most preferred algorithm for last two decades. A number of modifications to improve the security of RC4 have been reported in the literature (discussed in Sect. 4). However, the existing weaknesses reported in the year 2013 and 2014 on RC4 and its applications in WEP, WPA and TLS shows that the RC4 is an insecure algorithm and is still a challenging issue for research community.

4 Enhancements in RC4 Stream Cipher

Due to the RC4 weaknesses and related cryptanalytic attempts as discussed in Sect. 3, a number of variants of RC4 have been developed. Several enhancements of RC4 algorithm are presented in our survey article [15, 63]. A modified 32-bit RC4, named as *RC4 (n, m)* keystream generator, was proposed in [64, 65]. A modified RC4 KSA+ and PRGA+ with three layers of scrambling was proposed in [14]. Analysis of RC4+ illustrates that although the modified algorithm destroys many of the correlation between the state and the key but at the cost of encryption and decryption time. Run time of KSA+ and PRGA+ is 2.94 times (approx.) and 1.70 times than that of original RC4 KSA and RC4 PRGA respectively. Several new variants of RC4 including Quad-RC4, FJ-RC4, effective RC4, improved RC4, and RC4B were proposed in [66–73]. The available literature reveals that the RC4 variants proposed in the past were focused on eliminating the non-uniformity of bytes or the correlation between key and the state bytes. Also the proposed variants were targeted towards achieving better performance in terms of security, time or throughput. It is found that the existing proposals have changed the basic design of RC4, which is usually not desirable. Because the strength of RC4 lies in its robust design and simplicity.

However, in spite of numerous RC4 proposals, a number of challenges related to the searches of more biases, key collisions, and key recovery attack on WPA have not been explored till date. Therefore, there is a strong requirement to modify RC4 and to develop a new RC4 variant exhibiting better security and performance.

5 RC4 and its Variants

In the following section, we discuss the original RC4 and its various modified variants. We elaborate on the original RC4 [2], improved RC4 [65], RC4+ [14], FJ-RC4 [67] and effective RC4 [68]. The simulation parameters and attributes used for the implementation of RC4 and its variants are shown in Algorithm 1. The brief description of each existing RC4 algorithm is given below:

5.1 RC4

RC4 follows the design strategy used in stream ciphers. Pseudo code for RC4 KSA and PRGA is shown in Algorithm 2. Extracting pseudorandom data bytes from a pseudo-random permutation is the basic design principle of RC4 stream cipher. It has two working modules: the first is a Key Scheduling Algorithm (KSA) with key K as input (with typical size of 40–256 bits), and the second is Pseudo Random Number Generation Algorithm (PRGA), that generates a pseudo-random output sequence. The pseudo code for RC4 is presented in Algorithm 1. KSA generates 256 byte initial state vector S, by scrambling input state vector with a random key *K*. The S contains a permutation of 8 bit words i.e. 256 bytes. The secret key *K* is generally of length between 8 and 2048 bits and the expanded key *K* (length $N = 256$ bytes) is produced by performing simple repetitions. The expanded key is generated in the manner such that if secret key *k* is of length *l* bytes, the expanded key will be $K[i] = k\ [i \bmod L]$ for $0 \leq i \leq N-1$. Further, *S* pairs are swapped and an initial state S_{N-1} is achieved at the end, which is input to the second module PRGA. It generates the keystream of words from the permutation in S. Each iteration of the PRGA produces an output word, which is an output keystream byte. The generated byte is further xored with the plaintext to produce a ciphertext. It is to be noted that each time a new keystream byte *O* is required, RC4 runs the loop of PRGA and each time with the generation of new keystream, the internal state S is updated.

5.2 RC4+

The second RC4 variant that we have implemented is RC+. The pseudo code of RC4+ is shown in Algorithm 3. The cipher is modified as KSA+ and PRGA+. A three-layer architecture was proposed in KSA+ such that, the output keystream byte has no correlation with the secret key. Further, in PRGA+, masking was done in such a manner as if output byte is not coming directly from permutation byte. It was reported that the proposed algorithm diminishes known security attacks on RC4 including state recovery and distinguishing attacks. The running time of RC4 KSA+ and PRGA+ was claimed as 2.94 and 1.70 times, higher than that of RC4 KSA and PRGA respectively [14]. However, RC4 is widely accepted in numerous applications for its high speed. Though RC+ is providing high level of security, but at the cost of performance degradation in terms of time. The encryption time incurred in RC4+ is very high as compared to conventional RC4. So, there is a need for new RC4 implementation, which can overcome this tradeoff issue.

5.3 Improved RC4

The third implemented RC4 variant is an Improved RC4, pseudo code for which is presented in Algorithm 4. In an improved RC4, unlike conventional RC4 two S-boxes have been used which increase the randomness in the state and improve the statistical properties of the cipher. Authors have claimed that the proposed algorithm has removed many vulnerabilities of RC4 and the data can be encrypted with high speed of 0.875 s due to the parallel processing of output bytes.

5.4 FJ-RC4

The pseudo code of the fourth implemented variant, FJ-RC4 is given in Algorithm 5. In the proposed cipher the initial key was divided into three parts. Triple encryption and decryption is carried out to enhance the security of the cipher. Due to the occurrence of triple encryption and decryption, the authors claim the cipher to be more secure, but at the cost of running time, which is increased about three times as compared to plain RC4.

5.5 Effective RC4 Stream Cipher

Algorithm 6 presents the pseudo code of Effective RC4 stream cipher. In this cipher, KSA is the same as that in improved RC4. Modifications have been incorporated in PRGA. In PRGA, two parallel output bytes are generated, which are further xored with plaintext byte and the index j1 and j2 in two different steps. Due to parallel processing of output bytes and xoring of output byte with j index, the algorithm was claimed to be more secure and fast.

Algorithm 1. Simulation parameters used in RC4 and its variants

KSA **Initializations (for all implemented algorithms)** for i from 0 to 255 S[i] := i; end for *S1=S2=S; (When two states are considered)* j := 0; *j1=j2=j ;(When two states are considered)* L= length of the key N=length of the Substitution box or state key/key1/key2 = Key (Random numbers shared by the communicators)	**PRGA** **Initialization** i := 0; j := 0;
Simulation Attributes (for all the algorithms)	
State ([S])	256 bytes (state is of 256 bytes)
Key	16 bytes and will be expanded upto 256 bytes with mod N
Initialization Vector (IV)	16 bytes
Text size (Input)	1.25 million bytes *As RC4 is a stream cipher byte by byte processing is done. With each run of PRGA single byte of ciphertext is obtained* *All operations (addition) are under mod N condition*
All the simulations have been carried out in C language on *Intel i5, 2.5GHz, 2.2V and 28namp* machine.	

Algorithm 2. Pseudo code for RC4 stream cipher

KSA	PRGA
for i from 0 to N-1 j := (j + S[i] + key[i mod L]) ; swap (S[i], S[j]); end for	while message i := (i + 1); j := (j + S[i]); swap (S[i], S[j]); out O := S[(S[i] + S[j])]; end while

Algorithm 3. Pseudo Code for RC4+ Encryption Algorithm

KSA	PRGA
Layer 1 Basic scrambling for i from 0 to N − 1 j = (j + S[i] + K[i mod L]); swap(S[i], S[j]); **Layer 2 Scrambling with IV** for i =N/2 - 1 down to 0 j = (j + S[i]) xor (K[i mod L] + IV [i]); swap(S[i], S[j]); for i = 0, 1, 2, ….N − 1 j = (j + S[i]) xor (K[i mod L] + IV [i]); swap(S[i], S[j]); **Layer 3 Zigzag scrambling** for y = 0, . . . ,N − 1 if y ≡ 0 mod 2 then i=y/2; else i=N - (y+1)/2 j = (j + S[i] + K[i]); swap(S[i], S[j]);	while message i = i + 1; j = j + S[i]; swap(S[i], S[j]); t = S[i] + S[j]; t' = (S[i>>3 xor j<<5] + S[i<<5xor j>>3]) xor 0xAA; t"=j+S[j]; out O= (S[t] + S[t']) xor S[t"]; end while

Algorithm 4. Pseudo Code for Improved RC4 Encryption Algorithm

KSA	PRGA
for i from 0 to N-1 j1 := (j1 + S1[i] + key1[i mod K]) ; swap (S1[i], S1[j1]); j2 := (j2 + S2[i] + key2[i mod K]) ; swap (S2[i], S2[j2]); end for	while message i := i + 1; j1 := j1 + S1[i]; swap (S1[i] and S1[j1]); j2 := j2 + S2[i]; swap S2[i] and S2[j2]; out1 S1[S2[i] + S2[j2]]; out2 S2[S1[i] + S1[j1]]; swap (S1[S2[j1]], S1[S2[j2]]); swap (S2[S1[j1]], S2[S1[j2]]); end while

Algorithm 5. Pseudo Code for FJ-RC4 Encryption Algorithm [66]

```
Append zeros to make the length of the key divisible by 3

K=0;
  while main key
     Key1(K)=main key(i);
     Key2(K)=main key(i+1);
     Key3(K)=main key(i+2);
     K=K+1;
     i=i+3;
end

follow KSA and PRGA as in RC4
```

Algorithm 6. Pseudo Code for Effective RC4 Encryption Algorithm

```
PRGA
    while  message
    i=i+1;
    jl= jl+S1[i];
      swap(S1[i], S1[j]);
    j2= j2+S2[i];
      swap(S2[i], S2[j]);
out1= S1[(S1[i]+ S1[j1])] xor j1;
out2= S2[(S2[i]+ S2[j2])] xor j2;
      swap (S1[S2[j1]], S1[S2[j2]]);
      swap (S2[S1[j1]], S2[S1[j2]]);
end while
```

6 Proposed RC4 Algorithm

In Sect. 5, five different RC4 variants have been briefly described. It is found that, RC4+ is known to be the most secure cipher among all the existing variants, as it is resistant to many security attacks. However, security is achieved at the cost of performance degradation in terms of running time, almost thrice as compared to the conventional RC4. The RC4 modification proposed in improved RC4 provides high speed but low security as compared to RC4+. FJ-RC4 is again a weak cipher, where a plain RC4 runs three times, which results in performance degradation in terms of encryption time. Hence, it is not very efficient both in terms of security as well as encryption time. Looking into the security-performance tradeoff in existing variants, we have proposed three new implementations of RC4 referred to as; RC4-M1, RC4-M2 and RC4-M3. The proposed variants are based on

RC4+. To improve the run time, we have focused more on PRGA instead of KSA. It is because, during the whole encryption process, KSA runs only for one time to generate a state permutation for 256 bytes. However, PRGA runs every time to generate a single output byte. The simulation parameters and attributes used for the implementation of proposed algorithms are same as used for the implementation of existing algorithms and are shown in Algorithm 1.

6.1 RC4-M1

The first proposed variant RC4-M1 is based on RC4 KSA+. In the proposed cipher, three layer scrambling is performed in the same manner as with RC4+.

1. Initialization and first layer of scrambling is the same as the basic RC4 algorithm.
2. In the second layer, IV used is of same length as the secret key (256 bytes). The index i moves first from the middle down to the left end and then from the middle up to the right end. In our scheme, an *l*-byte IV, denoted by an array IV [0…l−1], is used from index N/2−1 down to N/2–l, during the leftward movement and the same IV is repeated from index N/2 up to N/2 + l–1, during the rightward movement.
3. In third and final layer of scrambling *i* takes values in the order: 0, 255, 1, 254, 2, 253…125, 130…128. Pseudo code for the RC4-M1 KSA and PRGA is shown in Algorithm 7 respectively.

The proposed variant RC4-M1 is different from RC4+ in a manner such that,

1. Two different keys K1 and K2 and two states S1 and S2 along with three layers of scrambling have been used. The keys—K1 and K2, throws S1 and S2 in confusion by generating two random permutations of {0, 1, 2…N–1}. The three layers of scrambling combined with two states remove correlation between secret key and permutation bytes, biases in different bytes, and chosen IV attacks to the great extent, which are the roots to several security attacks.
2. PRGA in RC4-M1, has the same structure as conventional RC4. Output stream is generated using S1 and S2, and two pseudo random words in one loop are obtained. In RC4-M1 PRGA two secret parameters j1 and j2 are obtained from S1 and S2 at the end of every loop, and four different secret states S1 [i], S1 [j1], S2 [i], S2 [j2] are computed. The elements of S1 swapped by S2 [i], S2 [j2] and the elements of S2 are swapped by S1 [i], S1 [j1]. As S1 [i], S1 [j1], S2 [i], S2 [j2] are the secret parameters, adversary will not come to know about which elements of S1 and S2 have been swapped. It also hides the relation between different bytes of S-boxes.
3. The proposed algorithm increases the internal states of S-boxes, which in turn increases the security of the cipher. In the proposed cipher security has been achieved but not at the cost of the performance of the cipher. Parallel processing in the proposed algorithm increases the speed of the algorithm. Pseudo code for the proposed PRGA is given in Algorithm 7.

Algorithm 7. Pseudo Code for Proposed RC4-M1 KSA Encryption Algorithm

KSA	PRGA
Layer 1 Basic scrambling for i from 0 to N − 1 j1 = (j1 + S1[i] + K1[i mod K]); swap (S1[i], S1[j1 mod K]); **j2= (j2+ S2[i] + K2[i]);** **swap (S2[i], S2[j2]);** end for **Layer 2 Scrambling with IV** for i =N/2 - 1 down to 0 j1= (j1+ S1[i]) xor (K1[i] + IV[i]); swap(S1[i], S1[j1]); end for for i = 0, 1, 2, ….N − 1 j1 = (j1 + S1[i]) xor (K1[i] + IV[i]); swap(S1[i], S1[j1]); end for for i =N/2 - 1 down to 0 **j2= (j2+ S2[i]) xor (K2[i] + IV[i]);** **swap(S2[i], S2[j2]);** end for for i = 0, 1, 2, ….N − 1 j2= (j2+ S2[i]) xor (K2[i] + IV[i]); swap(S2[i], S2[j2]); end for **Layer 3 Zigzag scrambling** for y = 0, . . . , N − 1 if y ≡ 0 mod 2 i=y/2; else i=N - (y+1)/2; end if j1= (j1 + S1[i] + K1[i]); swap(S1[i], S1[j1]); j2 = (j2 + S2[i] + K2[i]); swap(S2[i], S2[j2]); end for	while message i := i + 1; j1 := j1 + S1[i]; swap S1[i] and S1[j1]; j2 := j2 + S2[i]; swap S2[i] and S2[j2]; **out1 S1[S2[i] + S2[j2]];** **out2 S2[S1[i] + S1[j1]];** swap (S1[S2[j1]], S1[S2[j2]]); swap (S2[S1[j1]], S2[S1[j2]]); end while

6.2 RC4-M2

The second proposed modified RC4 variant is RC4-M2. As given in Algorithm 8, the proposed cipher is based on the design principle of KSA+.

1. In RC4-M2, KSA used is the same as KSA+.
2. Modification have been incorporated in RC4 PRGA+. Correlation between output byte and secret key, key recovery in IV mode, recovery of state permutation from generated output byte are some of the major weaknesses of RC4 PRGA. These weaknesses of PRGA were removed in PRGA+ by masking the output byte in a manner such that, it is not derived from any permutation byte.

Algorithm 8. Pseudo Code for Proposed RC4-M2 Encryption Algorithm

```
PRGA
while message
   i = i + 1;
   j = j + S[i];
    swap (S[i], S[j]);
t = S[i] + S[j];
t' = (S[i>>3 xor j<<5] + S[i<<5xor j>>3]) xor 0xAA;
t"=j+S[j];
   out1 z= (S[t] + S[t']) xor S[t"];
t' = (S[i>>3 xor j<<5] + S[i<<5xor j>>3]) xor 0x55;
t"=j+S[j];
   out2 z= (S[t] + S[t']) xor S[t"];
   end while
```

Algorithm 9. Pseudo Code for Proposed RC4-M3 Encryption Algorithm

```
PRGA
while message
   i := (i + 1);
   j := (j + S[i]);
   k := (j + S[i]+K2[i]);
    swap (S[i], S[j]);
   output := S[(S[i] + S[j]] xor S[k];
end while
```

3. In PRGA+ two permutation bytes are added in modulo 256, ((S [i ≫ 3 xor j ≪ 5] + S [i ≪ 5 xor j ≫ 3]) xor 0xAA)).
4. Further, to conceal the non-uniformity and to remove internal biases, the resultant of two added bytes is xored with third byte 0xAA, which is equivalent to 10101010. The addition of S [t'] and S [t''] in PRGA+ has increased its running time as compared to RC4 PRGA, which is not desirable in any real time applications.
5. To reduce the run time of PRGA+, it is modified such that, instead of generating single output, we generate two parallel outputs by adding one more layer in RC4-M2 PRGA ($t' = (S[i \gg 3 \text{ xor } j \ll 5] + S[i \ll 5\text{xor } j \gg 3]) \text{ xor } 0 \times 55$). Where the byte 0x55 represents 01010101, is xored with the two added bytes. It clearly shows that, generation of two parallel output bytes instead of one will reduce the run time of RC4-M2 PRGA as compared to PRGA+. Hence while maintaining the security of RC4+, RC4-M2 improves the performance as compared to RC4+. Security and performance analysis of all the proposed and existing RC4 variants is presented in Sects. 7 and 8 respectively.

6.3 RC4-M3

The third proposed RC4 variant is RC4-M3 in which, KSA is similar to the one used in KSA+ and RC4-M2, but with modified PRGA. Two different keys; K1 and K2 are used. The first key K1 is used in KSA and the second key K2 in PRGA.

1. As shown in Algorithm 9, one more layer of scrambling is added to the cipher *(k: = (j + S[i] +K2 [i]) mod 256).* The additional layer in PRGA includes the scrambling of permuted bytes with key K2 and a new index K is generated.
2. Further, to obtain the output byte, two bytes are added in module 256 and xored with index S[k] (*output: = S [(S[i] + S[j]) mod 256] xor S[k]).*
3. With this additional layer, we have enhanced the computational complexity of the cipher, intruder has to face the challenge of finding two different keys. The output byte is not directly dependent on keystream and the index *j*.

7 Security Analysis of RC4 Variants

The security of all the existing and the proposed variants has been evaluated through their respective security analysis on the basis of design, randomness analysis, and computational complexity in each variant.

7.1 Security Analysis

There are many flaws in traditional RC4 which makes it vulnerable to different security attacks. We have analyzed the security performance of proposed RC4 variants relative to the existing variants on the basis of their design structure. The security analysis of the existing variants and the three proposed RC4 variants is shown in Table 1. It is depicted that RC4+ is the strongest variant to date which resolves variety of security issues related to key recovery, state recovery and initial state biases while compromising the performance in terms of time offered by the cipher. We have proposed three new RC4 variants which are based on RC4+. In all the variants we have used RC4+ as their basic structure with some modification either in KSA+ or PRGA+ with the focus of either retaining or enhancing the security of the cipher. On the basis of the complexity offered by all the proposed variants we deduce that among all the proposed RC4 variants security provided by RC4-M2 > RC4-M1 > RC4-M3. Among all the implemented existing and proposed variants security provided by RC4-M2 > RC4+ > RC4-M1 > RC4-M3 > FJ-RC4 > Improved RC4 > Effective RC4. We have theoretically analyzed the resistance of proposed ciphers against cryptanalytic attempts.

7.1.1 Brute Force Attack

Brute force is an exhaustive key search attack. Intruder tries each and every possible combination of key to find the plaintext. The proposed schemes RC4-M1 and RC4-M3 are resistant to brute force attack as key length has increased from 256 to 512 bytes by using of two different keys.

Table 1 Security analysis of RC4 variants

RC4 variants	Design of RC4 variants	Security analysis
RC4	This is a basic original RC4 structure	Many flaws have been reported in conventional RC4 as discussed in Sect. 3 [16–62]
Improved RC4	Based on conventional RC4, but increased the state space twice the original RC4 i.e. from 1700 to 3400 by including two S boxes and two keys	It is hard for intruder to find two keys Computational complexity is increased which makes the algorithm resistant against brute force attack Statistical properties are improved using double permutation [66] But not as secure as RC4+ [14]
FJ-RC4	Based on conventional RC4, Key is divided into three parts, triple encryption and decryption is performed. Both KSA and PRGA are same as original RC4	Simply increased the computational complexity of the algorithm [68] No improvement in the statistical properties of the algorithm which are the roots to several attacks on RC4
Effective RC4	Is a combination of plain RC4 and improved RC4, KSA and PRGA are similar to improved RC4 except the step *Output = M[x] xor Generated Key1 xor j1;* *Output = M[x] xor Generated Key2 xor j2;* where two outputs are generated and xored with index j1 and j2	Computational complexity is increased which makes the algorithm resistant against brute force attack [69] Statistical properties are improved using double permutation and xor operation with index j1 and j2 But not as secure as RC4+ [14]
RC4+	Improvements are done in both KSA and PRGA KSA+: three layer scrambling is done Initial scrambling is same as plain RC4 Layer two scrambling is done using IV Zig zag scrambling is performed by moving each adjacent byte PRGA+: Shifting and xoring of bytes is done to increase the dependency of single byte upon many bytes. Xor with 0xAA is done	Removes several initial byte biases [14] Xor operation helps to remove out many biases Zig Zag scrambling removes the relationship between key bytes and the permutation bytes Hides the relation between output keystream and the secret key Resistant to state recovery attacks Resistant to distinguishing attacks (*RC+ is the most secure algorithm among all the implemented existing algorithms also among the algorithms reported till date*)
RC4-M1	Combination of RC4+ and improved RC4 Three layer scrambling is used same as KSA+ State space and randomness is further increased using two states S1 and S2, and two keys K1 and K2 Two indices j1 and j2 are used and two parallel outputs are obtained	Offers similar security as provided by RC4 KSA+ along with improved performance in terms of time Make the task of intruder more difficult by offering him double challenge of increased state space and two keys Like KSA+, improves the randomness properties of the cipher and remove many initial state biases hence resistant to several attacks reported in the literature Use of double state and indices make the algorithm resistant to state recovery attacks

Table 1 continued

RC4 variants	Design of RC4 variants	Security analysis
RC4-M2	Based on RC4+ KSA used is same as KSA+ PRGA+ is modified to further increase the security and performance as well Two parallel outputs are generated by adding one more layer ***(t′ = (S[i ≫ 3 xor j ≪ 5] + S[i ≪ 5xor j ≫ 3]) xor 0x55)***	Proposed algorithm addresses all issues that have been resolved by RC4+ Removes all the vulnerabilities of existing ciphers More secure than PRGA+, rather complete RC4+ Non-uniformity is concealed by two different bytes instead of one, unlike PRGA+ output depends on two additional bytes Performance in terms of time is better than RC4+
RC4-M3	KSA is similar to KSA+ PRGA of plain RC4 is used with some modification Two different keys are used, one in KSA and another in PRGA A new byte k is generated using K2, which is further xored with the output byte	KSA is as secure as KSA+ Use of K2 makes the PRGA secure Output byte is not directly coming from permutation byte Makes the state recovery attack difficult Improved performance in terms of time

7.1.2 Permutation Recovery Attack

The basic idea of permutation recovery attack is elaborated in [46], where permutations can be recovered easily. The proposed schemes are resisting permutation recovery attacks, in PRGA, output keystream $[S^G[S^G[i^G] + S^G[j^G]]$ is not directly derived from the state permutation, instead is masked by various other operations. For the cryptanalysis one is required to first estimate $S^G[t]$, $S^G[t']$, and $S^G[t'']$. To find the output value, intruder has no option, than to go for all the probable choices. The inclusion of additional operations in PRGA, t' and t'', ensures the non-recovery of RC4 permutation from the output keystream byte and the idea of [46] will not work.

7.1.3 Distinguishing Attacks

The distinguishing attack challenges the pseudorandom generation of bytes in the stream cipher, a basic claim of any stream cipher. This type of attack initiates from the fact that when second state byte is 0 and first byte is not equal to 2, the second output byte will take the value of 0. Many cryptanalysis attempts have been made on the basis of this fact [53]. In the proposed schemes, output keystream is generated in different manner as compared to RC4 PRGA. It does not produce a non-random output and make the cipher free from such biases.

7.1.4 Key Correlation Attacks

The attack aims at finding any correlation between output keystream and the secret key and leads to key recovery attacks. In the proposed schemes the index i is moving first from middle bytes to left end enables the swapping of bytes in the first quarter of permutation,

that were in linear combination with secret key bytes. It results in the removal of initial byte biases. Similar operation is performed on the second half and removes the biases at the time of inverse permutation. Further the use of xor operation is also eliminating these biases. Also the zig-zag scrambling occurring in layer 3 of KSA is preventing the formation of recursive equation [24] and hiding the connection between key and permutation bytes. So it is deduced that the proposed schemes are resistant to key recovery attacks as there is no correlation between key bytes and secret key bytes.

7.1.5 Chosen IV Attack

In the conventional schemes, the improper use of initialization vector makes the cipher vulnerable to IV-mode attacks [32]. The IV's were either prefixed or suffixed with secret key. In our proposed variants, the IV is used in the middle and added with key bytes in each iteration during the updation of index *j*. Moreover, IV is involved only in Layer 2, and not used in layer 3 where zig-zag scrambling is involved. This step helps the cipher to get rid of chosen IV vector attack.

7.2 Randomness Analysis

It is always desirable that output of PRGA must be unpredictable without the knowledge of any input. In particular, without knowing the key, intruder must not be able to develop the present or future messages, even if he had gained an access to any previously generated random sequence. There should be no correlation between the key and the generated output sequence. We have proposed the three different RC4 variants to increase the randomness which in turn increase the security of the cipher as compared to the existing RC4 variants. To analyze the security of different RC4 variants we have studied the degree of randomness associated with each cipher. To investigate the degree of randomness offered by RC4 PRGA, we have performed extensive experimentation with the NIST statistical test suite. We have opted NIST test suite for its accuracy and popularity. The NIST framework, based on hypothesis testing, is a set of 15 statistical tests to examine the randomness of binary sequences (a long sequence) generated by any PRGA. The NIST statistical test suit emphasis on a number of different type of non-randomness that could exist in any binary sequence.

Each test has been designed to detect specific type of flaws. The different tests and their general characteristics are shown in Table 2. Using NIST STS for RC4, we have investigated, whether or not the generated sequence of zeros and ones is random. We have implemented all the variants and statistical tests in MATLAB 13. We have conducted our experiments on over 10 lac bytes. All the generated RC4 PRGA sequences are applied to the NIST STS and the result of each test was analyzed to decide whether or not it passes the randomness tests. A generated random sequence to be accepted or rejected is decided by comparing the p value to 0.01.

If p value is more than 0.01, the sequence is random and is accepted else the sequence is non-random and is rejected. If the generated sequence will pass all the statistical tests, only then it will be concluded that the resultant sequence is truly random and RC4 can be securely used in wireless networks. p Values obtained after implementing all the 15 statistical tests are shown in Table 3, where 'Success' indicates that for all implemented RC4 variants, the obtained p values are >0.01. All the algorithms are passing the NIST

Table 2 NIST statistical test suite

	NIST statistical test	Test purpose	Flaws detected
1	Frequency (Monobit) Test	Probability of the occurrence of number of 0's and 1's should be approximately same i.e. 0.5	Number of 0's is more than number of 1's or vice versa
2	Frequency test within a block	To determine whether the number of 1's in P bit block is approximately P/2	Too many 0's or 1's in P bit block
3	Runs Test	Focus on the total number of runs in the sequence. Determine whether the number of runs of o's and 1's of different length is as desired for a random sequence	More or less total runs show that the oscillation between the bit streams is too fast or too slow
4	Test for the longest Run of 1's in a Block	The test is focused on whether the length of the longest run of 1's in the tested sequence is as expected in a random sequence	Irregularity in the distribution of longest run of 1's within the P-bit block
5	Binary Matrix Rank Test	The purpose of the test is to check for the linear dependence among fixed length sub-strings of the actual sequence by finding the rank of the disjoint sub-matrices of the whole sequence	Irregularity in the rank distribution from consistent random sequence
6	Discrete Fourier Transform Test/ Spectral Test	The purpose of the test is to detect the repetitive patterns that are close to each other of the tested sequence	Periodic features of the bit stream
7	Non-overlapping Template Matching test	Occurrence of aperiodic pattern in the sequence using non-overlapping p-bit window	Very large occurrences of aperiodic templates
8	Overlapping Template Matching test	Occurrence of aperiodic pattern in the sequence using overlapping p-bit window	High occurrence of p-bit runs of 1's
9	Maurer's Universal Statistics Test	To check whether the sequence can be compressed without losing any information	Compressibility reflects the non-random behavior of the sequence
10	Linear Complexity Test	To find the length of the linear feedback shift register (LFSR) and to determine whether the sequence is complex enough to be considered as random	Longer LFSR and less complex sequence imply better randomness
11	Serial Test	To determine that every m-pattern is getting the same chance of appearing as every other pattern	Non-uniformity in the distribution of patterns
12	Approximate Entropy Test	The purpose of the test is to compare the frequency of all possible overlapping blocks of two adjacent lengths (m and m + 1)	Non-uniformity in the distribution of patterns
13	Cumulative Sums test/Cusum Test	Whether the cumulative sum of the test sequence is very large or small as compared to the expected behavior of the random sequence	More number of 0's or 1's at the beginning of the sequence
14	Random Excursions Test	Determine if the number of visits to a particular state within a cycle varies from the one that would be expected from a random sequence	Irregularity in the distribution of the number of visits

Table 2 continued

	NIST statistical test	Test purpose	Flaws detected
15	Random Excursions Variant Test	To determine the total number of times the event is occurred in a cumulative sum random walk	Deviation from the expected number of occurrences

statistical test suite; hence the generated PRGA keystream is truly random and uniformly distributed.

7.3 Computational Complexity (CC)

In this paper, we have analyzed the computational complexity in terms of the number of operations (N_o) incurred in each RC4 variant. Computational complexity of each cipher is shown in Table 4. CC is evaluated by finding out the total number of operations incurred in both KSA and PRGA. More the number of operations more will be the complexity of the cipher. Complexity of each variant is analyzed as below:

$$\text{In basic RC4 in} \quad \text{KSA} \quad N_o = 3 \times 256 = 768$$

where 256 is the total number of bytes in the state box S and

$$\text{PRGA} \quad N_o = 6 \times N = 6N \text{ bytes}$$

N represents the number of plaintext bytes. For example if $N = 40$ then number of iterations in PRGA will be 40 and the number of operations will be $6 \times 40 = 240$. So the computational complexity of

$$\text{Basic RC4} \quad CC = 768 + 6N(\text{Total number of operations in KSA} + \text{PRGA})$$

Similarly we have evaluated total number of operations for each RC4 variant and calculated the complexity associated with each cipher.

For

$$\begin{array}{ll} \text{Improved RC4} & CC = 1536 + 14N \\ \text{FJ-RC4} & CC = 2304 + 18N \\ \text{Effective RC4} & CC = 1536 + 14N \\ \text{RC4+} & CC = 4608 + 16N \\ \text{RC40M1} & CC = 9216 + 14N \\ \text{RC40M2} & CC = 4608 + 29N \\ \text{RC40M3} & CC = 4608 + 8N \end{array}$$

It is interpreted that complexity associated with basic RC4 is minimum ($768 + 6N$) which leads for better time and poor security performance. Though CC in improved RC4, effective RC4 and proposed RC4-M1 PRGA are same (14N) but they vary in their security performance due to the complexity of KSA which is very high in RC4-M1 (CC = 9216). CC of FJ-RC4 are thrice that of basic RC4 ($2304 + 18N$). In this case with increase in CC, security is enhanced with degraded time performance. CC in the case of RC4+, RC4-M2 and RC4-M3 is $4608 + 16N$, $4608 + 29N$, $4608 + 8N$. As the complexity associated with

Table 3 Randomness test results

Test no.	Statistical tests	Calculated p values								
		RC4	RC4 KSA+	Improved RC4	RC4 FJ	Effective RC4	RC4 M1	RC4 M2	RC4 M3	Conclusion
1	Frequency (Monobit) Test	0.3153	0.1048	0.1795	0.9426	0.2135	0.8243	0.8571	0.0280	Success
2	Frequency test within a block	1	1	1	1	1	1	1	1	Success
3	Runs Test	0.8616	0.8615	0.6710	0.5688	0.6136	0.2145	0.5963	0.2134	Success
4	Test for the longest Run of 1's in a Block	0.9557	0.8503	0.6601	0.0162	0.8557	0.6044	0.1830	0.6830	Success
5	Binary Matrix Rank Test	0.8654	0.8615	0.7521	0.5012	0.8543	0.6521	0.4628	0.5125	Success
6	Discrete Fourier Transform Test/Spectral Test	0.7134	0.7541	0.5124	0.249	0.7234	0.7524	0.2148	0.6521	Success
7	Non-overlapping Template Matching test	0.3333	0.6313	0.4647	0.9801	0.3987	0.8778	0.7532	0.5774	Success
8	Overlapping Template Matching test	1	1	1	1	1	1	1	1	Success
9	Maurer's Universal Statistics Test	0.0186	0.0186	0.0186	0.0186	0.0186	0.0185	0.0186	0.0186	Success
10	Linear Complexity Test	0.9842	0.9661	0.9257	0.9817	0.9728	0.8726	0.8728	0.9883	Success
11	Serial Test	0.3451	0.2216	0.9867	0.9363	0.3952	0.9736	0.9869	0.7515	Success
12	Approximate Entropy Test	0.0257	0.2774	0.5587	0.3972	0.0226	0.9181	0.9992	0.2815	Success
13	Cumulative Sums test/Cusum Test	0.2804	0.1697	0.3394	0.9981	0.2527	0.7270	0.2570	0.0464	Success
14	Random Excursions Test	0.7346	0.4909	0.1919	0.1657	0.2683	0.5617	0.1859	0.3735	Success
15	Random Excursions Variant Test	0.9996	0.9995	1	0.9988	0.9942	0.9987	0.9994	0.9962	Success

Table 4 Analysis of computational complexity

RC4 variants	Algorithm	Type of operation					Number of operations (NO) in KSA and PRGA	Computational complexity (C)
		XOR	SWAP	ADD	MOD	SHIFT		
RC4	KSA	0	1	1	1	0	$3 \times 256^* = 768$	$768 + 6N$
	PRGA	1	1	2	2	0	$6 \times N^{**}$ bytes	
Improved RC4	KSA	0	2	2	2	0	$6 \times 256 = 1536$	$1536 + 14N$
	PRGA	2	4	4	4	0	$14 \times N$ bytes	
FJ-RC4	KSA	0	3	3	3	0	$9 \times 256 = 2304$	$2304 + 18N$
	PRGA	3	3	6	6	0	$18 \times N$ bytes	
Effective RC4	KSA	0	2	2	2	0	$6 \times 256 = 1536$	$1536 + 14N$
	PRGA	2	4	4	4	0	$14 \times N$ bytes	
RC4+	KSA+	2	4	6	6	0	$18 \times 256 = 4608$	$4608 + 16N$
	PRGA+	4	1	4	3	4	$16 \times N$ bytes	
RC4-M1	KSA	4	8	12	12	0	$36 \times 256 = 9216$	$9216 + 14N$
	PRGA	2	4	4	4	0	$14 \times N$ bytes	
RC4-M2	KSA	2	4	6	6	0	$18 \times 256 = 4608$	$4608 + 29N$
	PRGA	8	1	8	4	8	$29 \times N$ bytes	
RC4-M3	KSA	2	4	6	6	0	$18 \times 256 = 4608$	$4608 + 8N$
	PRGA	1	1	3	3	0	$8 \times N$ bytes	

* 256 represents the total number of state bytes

** N represents the length of plain text

RC4-M2 is the maximum, it provides maximum security with little bit high execution time but less than RC4+.

8 Performance Analysis

RC4 is a stream cipher, where byte-by-byte processing is performed under MOD N condition. Key size and state length are the two important attributes of this stream cipher. In the present work, Key size is considered of 16 bytes and state box of 256 bytes. In each round a single byte is generated after PRGA, which is xored with single input byte to generate a cipher text byte. Performance analysis of all the RC4 variants has been carried out in terms of running time, CPU cycles, energy cost and throughput. All the simulations have been carried out in C language on *Intel i5, 2.5 GHz, 2.2 V and 28namp* machine. We have encrypted 1.25 million bytes with all the variants.

8.1 Run Time

Run time is the time taken by any cipher in encryption or decryption of data. Table 5 and Fig. 1 present the run time analysis of all the RC4 variants along with their security performance tradeoff. We have evaluated the time consumed in generating 1.25 million output bytes from PRGA for each cipher. In this case, key size does not affect the performance of the cipher. It is used only once in KSA and not used in PRGA. Moreover, if

Table 5 Performance analysis of RC4 variants

RC4 variant	Run time		CPU cycles (Mcycles)	Energy(J)	Throughput (Mbps)	Security and performance tradeoff analysis
	KSA (μsec)	PRGA (sec)	PRGA	PRGA	PRGA	
RC4	18.3	0.062	155	9.548	20.161	Very fast, efficient and simple in design but not a secure cipher
Improved RC4	23	0.109	272.5	16.786	11.467	Security is increased due to increased state space as compared to plain RC4 with poor performance in terms of time, cycles, energy and throughput
FJ-RC4	54.9	0.17	425	26.18	7.352	Time is almost thrice as compared to plain RC4 because of triple encryption and decryption result in poor performance
Effective RC4	23	0.110	280	17.248	11.160	Time consumed is slightly greater than improved RC4 due to the additional xor operation
RC4+	43.7	0.156	390	24.024	8.012	Running time of RC4+ is very high as compared to plain RC4 and its improved variants due to shifts and xor operations. Security is achieved at the cost of performance
RC4-M1	64.9	0.109	272.5	16.786	11.467	Security provided is more than Improved RC4 with similar execution time Security provided by KSA is similar to KSA+ Time consumed is less than RC4+ because of the simple structure of PRGA
RC4-M2	43.7	0.14	350	21.56	8.928	Run time is less than RC4+ due to the generation of two parallel outputs Security is achieved without compromising the performance of the cipher
RC4-M3	43.7	0.08	200	12.32	15.625	Time consumed is slightly greater than plain RC4 but less than all the existing and proposed variants

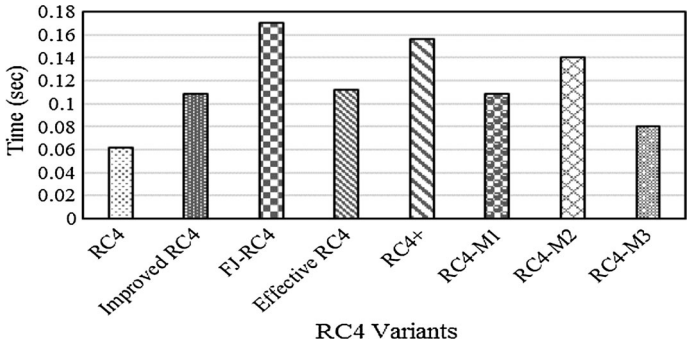

Fig. 1 Running time comparison of RC4 variants

we keep both the key size and the state [S] smaller, it will reduce the security of cipher. The size of text affects the encryption time in PRGA. Larger will be the text, more number of times the PRGA will run resulting in greater time consumption.

We have computed the time consumed by KSA in generating the new keystream. KSA runs only once, hence it will not affect the performance of RC4 to large extent. PRGA will run with the length of plaintext to generate the ciphertext bytes. Every time it will be incremented to produce the output byte. It is observed that the time incurred is the minimum for plain RC4 due to its design simplicity and is the least secure algorithm. As we increase the security of the cipher, encryption time increases.

Though the execution time of improved RC4 and effective RC4 is not very high but these algorithms are not resistant to many security attacks as compared to RC4+. The RC4+ is the most secure algorithm but time consumption is almost thrice as compared to the basic RC4. We have proposed three RC4 variants while focusing on this security performance tradeoff. Figure 2 presents the comparative analysis of conventional RC4, RC4+ and the proposed RC4 variants. It demonstrates that the execution time incurred in RC4+ is 60, 30, 10, and 48.7 % higher than conventional RC4, RC4-M1, RC4-M2 and RC4-M3 respectively. In the proposed variants, encryption time is decreased as compared to RC4+ and reflects the computational efficiency of proposed schemes. In RC4-M1 and RC4-M2, two parallel outputs are obtained, hence reduces the time consumption. In RC4-M3 PRGA is based on plain RC4 with the additional byte that is generated using K2 and further xored with output byte and plain text which increase the security of the cipher without compromising the performance.

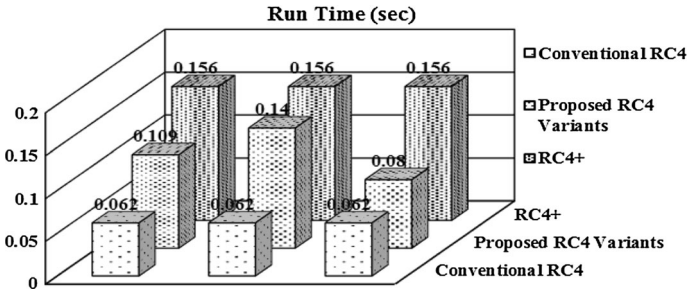

Fig. 2 Comparative analysis of run time of conventional RC4, RC4+ and proposed RC4 variants

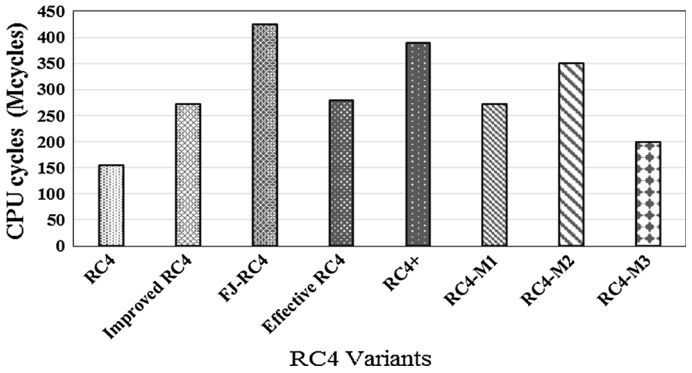

Fig. 3 CPU cycles consumed in various RC4 variants

8.2 CPU Cycles

It measures the number of clock cycles incurred in encrypting 1.25 million bytes with various RC4 variants. Table 5 and Fig. 3 present the results for the number of clock cycles. It is observed that similar trends as with run time are followed for clock cycles. To be more specific we have presented the encryption speed of all the RC4 variants in terms of cycles/byte in Table 6. The encryption speeds for RC4+, RC4-M1, RC4-M2 and RC4-M3 are 312 cycles/byte, 218 cycles/byte, 280 cycles/byte and 160 cycles/byte respectively. Speed of the proposed schemes—RC4-M1, RC4-M2 and RC4-M3 PRGA are 1.43, 1.11 and 1.95 times greater than RC4+ PRGA+ respectively.

8.3 Energy Cost

It is the amount of energy consumed during encryption process. Energy consumption can be measured by counting the number of cycles incurred in the encryption process. Energy cost is determined by the number of cycles, operating voltage of CPU, and average current drawn for each cycle in accordance with the procedure followed in [70]. It is given by

$$\mathrm{E} = \mathrm{Vcc} \times I \times \mathrm{N} \tag{1}$$

where E is the energy cost, Vcc is the supply voltage of the system (2.2 V), N is the number of cycles, and I is the average current drawn from the power source (28 namp). Typically, for 155 Mega cycles consumed in the plain RC4 is computed as

Table 6 Cycles/byte consumed in RC4 variants

RC4 variant	Cycles/byte
RC4	124
Improved RC4	218
FJ-RC4	340
Effective RC4	224
RC4+	312
RC4-M1	218
RC4-M2	280
RC4-M3	160

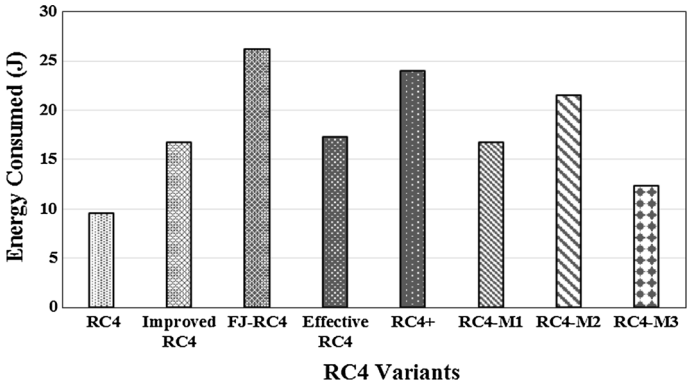

Fig. 4 Energy consumption in RC4 variants

$$E = 155\text{Mcycles} \times 2.2\text{V} \times 28\text{namp} = 9.548\text{J} \quad (2)$$

This operation is used to evaluate the energy cost for all the existing and proposed RC4 variants. As shown in Fig. 4, the energy cost of RC4 variants varies in the order of RC4+ > RC4-M2 > RC4-M1 > RC4-M3 > RC4 as depicted in the Table 5. It shows that RC4+ consumes the maximum energy due to the highest time incurred in the encryption/decryption process and the energy cost of the proposed algorithms is less than RC4+.

8.4 Throughput

It is the amount of data bytes encrypted per second, measured in Mbps. Throughput obtained varies as RC4+ < RC4-M2 < RC4-M1 < RC4-M3 < RC4 as depicted in Fig. 5 and Table 5.

8.5 Performance Comparison of RC4 and HC-128

Performance analysis of HC-128 and conventional RC4 is given in the Table 7, which shown that RC4 outperforms HC-128 for all the considered performance measures.

In this paper we have carried out the performance analysis of some of the recently proposed existing variants and compared with the proposed RC4 variants. The results

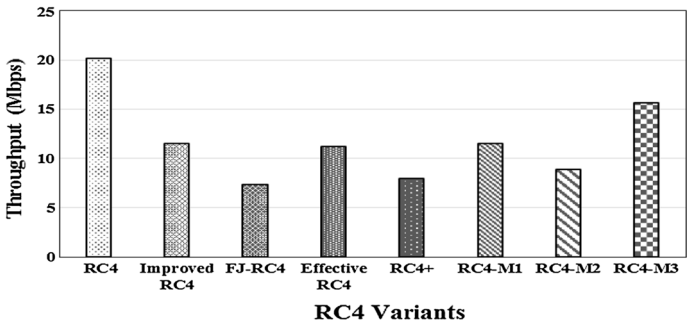

Fig. 5 Throughput comparison of RC4 variants

Table 7 Performance comparison of RC4 and HC-128

Stream cipher	Run time (s)	CPU cycles (MCycles)	Energy(J)	Throughput (Mbps)	cycles/byte
RC4	0.062	155	9.548	20.161	124
HC-128	0.075	187.5	11.55	16.6	150

presented in the paper demonstrate that security and network performance work in contrast to each other. The simulation results presented in this paper proves that the proposed RC4 variants results in better security performance tradeoff as compared to the existing variants. Hence the proposed algorithms outperform the existing variants in terms of both security and performance.

9 Recommendations for Potential Applications

Our simulation results and their analysis identify the appropriate security algorithm in the given network scenario. An acceptable level of both security and its associated performance is always required for designing any application. Application designers always have different inclinations, on the basis of the risk they can tolerate and the performance price.

For instance, in conversational services for audio and video conversation, security can be compromised but not the speed [74, 75]. In these applications, high throughput is always desired. More complex RC4 variants cannot be used in such scenarios, because increase in complexity will decrease the throughput. In such a scenario, RC4-M3, providing high speed without compromising the security to large extent is recommended.

In interactive services low throughput and high run time greater than conversational services can be tolerated. For transaction services high security even at the cost of performance is required. For such scenarios, RC4-M2, due to its high security is recommended as the proposed variant provides high security without incurring any transactional delays. Streaming services require high data rate values. On the basis of the sensitivity of data and user requirements RC4-M1 and RC4-M2 can be used in such applications.

Background services, also known as best effort services do not have any specific performance constraints. Depending on the security requirements any of the proposed variants RC4-M1, RC4-M2 and RC4-M3 can be implemented in these services. The numerical results presented in this paper can be used to choose a security algorithm, depending upon the sensitivity of data transmitted and the performance requirements by users.

10 Conclusion

In this paper, we have presented different implementations of RC4 stream cipher and elaborated on the various weaknesses in the existing variants. Further, we have proposed three RC4 variants by introducing additional layers of scrambling in the existing variants. The security of the proposed algorithms is tested by performing their randomness analysis using NIST STS, which is a package of 15 randomness analysis tests. It is proved that the proposed ciphers conform to all these randomness tests. Secondly the security of the proposed variants on the basis of their design structure has been analyzed and proved that

the proposed algorithms are offering more challenges and complexity to the intruder as compared to the existing RC4 variants. Resistance of proposed schemes against several cryptanalytic attempts is discussed and justified their strength in removing these attacks. Further, performance analysis of all the RC4 variants has been carried out in terms of run time, CPU cycles, energy cost and throughput. It is observed that while providing the comparable security, the running time of RC4-M1, RC4-M2 and RC4-M3 is 30, 10, and 48.7 % lower than that of RC4+. Similar variations are observed for number of cycles and energy cost. The throughput achieved with proposed variants is high as compared to RC4+ and other variants. These results show that computation load of the proposed variants as compared to the RC4+ is significantly reduced, concluding that the proposed schemes are computationally efficient.

Due to computational complexity of the proposed algorithms, the execution time incurred is higher than the basic RC4, which corroborates the fact that there is always a tradeoff between security and network performance. This tradeoff is optimized in the proposed RC4 variants. The performance analysis presented herein may be used as reference for selecting the particular RC4 variant for given applications/service as required. The analysis presented in this paper shows that there is a tradeoff between security and network performance and efforts can be made to further optimize this tradeoff.

References

1. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Hand-book of applied cryptography* (2011th ed.). Boca Raton: CRC Press **(fifth printing)**.
2. Stinson, D. R. (1995). *Cryptography: Theory and practice* (2005th ed.). Boca Raton: CRC Press.
3. Biryukov, A., Shamir, A., & Wagner, D. (2000). Real time cryptanalysis of A5/1 on a PC. In B. Schneier (Ed.), *FSE, volume 1978 of lecture notes in computer science* (pp. 1–18). New York: Springer.
4. Bluetooth, T. M. (2010). *Bluetooth specification, v4.0*. E0 encryption algorithm described in volume 2, pp. 1072–1081. http://www.bluetooth.org.
5. Briceno, M., Goldberg, I., & Wagner, D. (1998). *A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms*. http://www.scard.org/gsm/a51.html.
6. Third Generation Partnership Project. (2006). *Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2*. ETSI/SAGE Specification Document 2: SNOW 3G Specification, v1.1, pp. 1–27. ESTI/SAGE Specifications.
7. ECRYPT Stream Cipher Project eSTREAM. *The current eSTREAM portfolio*. http://www.ecrypt.eu.org/stream/index.html.
8. ECRYPT Stream Cipher Project eSTREAM. *Software performance results from the eSTREAM project*. http://www.ecrypt.eu.org/stream/perf/#results.
9. Rivest, R. L. (2001). *RSA security response to weaknesses in key scheduling algorithm of RC4*. Technical note, RSA Data Security, Inc.
10. Nawaz, Y., Gupta, K. C., & Gong, G. (2005). A 32-bit RC4-like keystream generator. *IACR Cryptology ePrint Archive, 2005*, 175.
11. Biham, E., & Seberry, J. (2006). Pypy: Another version of Py. *eSTREAM, ECRYPT Stream Cipher Project, Report, 38*, 2006.
12. Biham, E., & Seberry, J. (2007). *Tweaking the IV setup of the Py family of stream ciphers—The ciphers TPy, TPypy, and TPy6*. http://www.cs.technion.ac.il/biham/. Accessed 25 Jan 2007. 2, 4.
13. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., & Barker, E. (2001). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. Mclean: Booz-Allen and Hamilton Inc Mclean Va.
14. Maitra, S., & Paul, G. (2008). Analysis of RC4 and proposal of additional layers for better security margin. In D. R. Chowdhury, V. Rijmen, A. Das (Eds.), *Progress in cryptology INDOCRYPT 2008* (pp. 27–39). Springer: Berlin.
15. Jindal, P., & Singh, B. (2015). RC4 encryption-A literature survey. *Procedia Computer Science, 46*, 697–705.

16. Roos, A. (1995). *A class of weak keys in the RC4 stream cipher*. Two posts in sci.crypt, message-id 43u1eh$1j3@hermes.is.co.za and 44ebge$llf@hermes.is.co.za, http://www.impic.org/papers/WeakKeys-report.pdf.
17. Paul, G., Rathi, S., & Maitra, S. (2008). On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. *Designs, Codes and Cryptography, 49*(1–3), 123–134 **(initial version in proceedings of WCC 2007)**.
18. Wagner, D. A. (1995). *My RC4 weak keys*. Post in sci.crypt, messageid 447o1 l$cbj@cnn.Princeton.EDU. http://www.cs.berkeley.edu/∼daw/my-posts/my-rc4-weak-keys.
19. Grosul, A. L., & Wallach, D. S. (2000). *A related-key cryptanalysis of RC4*. Technical Report TR-00-358, Department of Computer Science, Rice University.
20. Biham, E., & Dunkelman, O. (2007). Differential cryptanalysis in stream ciphers. *IACR Cryptology ePrint Archive, 2007*, 218.
21. Matsui, M. (2009). Key collisions of the RC4 stream cipher. In O. Dunkelman (Ed.), *FSE, volume 5665 of lecture notes in computer science* (pp. 38–50). New York: Springer.
22. Chen, J., & Miyaji, A. (2011). How to find short RC4 colliding key pairs. In X. Lai, J. Zhou, & H. Li (Eds.), *ISC volume 7001 of lecture notes in computer science* (pp. 32–46). Springer: New York.
23. Maitra, S., Paul, G., Sarkar, S., Lehmann, M., & Meier, W. (2013). New results on generalization of roostype biases and related keystreams of RC4. In A. Youssef, A. Nitaj, & A. E. Hassanien (Eds.), *AFRICACRYPT, volume 7918 of lecture notes in computer science* (pp. 222–239). New York: Springer.
24. Paul, G., & Maitra, S. (2007). Permutation after RC4 key scheduling reveals the secret key. In C. M. Adams, A. Miri, & BIBLIOGRAPHY M. J. Wiener (Eds.), *Selected areas in cryptography, volume 4876 of lecture notes in computer science*, (pp. 360–377). Springer.
25. Biham, E., & Carmeli, Y. (2008). Efficient reconstruction of RC4 keys from internal states. In K. Nyberg (Ed.), *FSE, volume 5086 of lecture notes in computer science* (pp. 270–288). New York: Springer.
26. Akgün, M., Kavak, P., & Demirci, H. (2008). New results on the key scheduling algorithm of RC4. In D. Chowdhury, V. Rijmen, & A. Das (Eds.), *INDOCRYPT, volume 5365 of lecture notes in computer science* (pp. 40–52). New York: Springer.
27. Khazaei, S., & Meier, W. (2008). On reconstruction of RC4 keys from internal states. In J. Calmet, W. Geiselmann, & J. Müller-Quade (Eds.), *MMICS, volume 5393 of lecture notes in computer science* (pp. 179–189). New York: Springer.
28. Basu, R., Maitra, S., Paul, G., & Talukdar, T. (2009). On some sequences of the secret pseudo-random index *j* in RC4 key scheduling. In M. Bras-Amorós & T. Høholdt (Eds.), *AAECC, volume 5527 of lecture notes in computer science* (pp. 137–148). New York: Springer.
29. Fluhrer, S. R., Mantin, I., & Shamir, A. (2001). Weaknesses in the key scheduling algorithm of RC4. In S. Vaudenay & A. M. Youssef (Eds.), *Selected areas in cryptography, volume 2259 of lecture notes in computer science* (pp. 1–24). New York: Springer.
30. Korek. (2004). *Need security pointers*. http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036.
31. Korek. (2004). *Next generation of WEP attacks?* http://www.netstumbler.org/showpost.php?p=93942&postcount=%35.
32. Mantin, I. (2005). A practical attack on the fixed RC4 in the WEP mode. In B. K. Roy (Ed.), *ASIACRYPT, volume 3788 of lecture notes in computer science* (pp. 395–411). New York: Springer.
33. Klein, A. (2008). Attacks on the RC4 stream cipher. *Designs, Codes and Cryptography, 48*(3), 269–286 **(published online in 2006, and accepted in WCC 2007 workshop)**.
34. Tews, E., Weinmann, R.-P., & Pyshkin, A. (2007). Breaking 104 bit WEP in less than 60 seconds. In S. Kim, M. Yung, & H.-W. Lee (Eds.), *WISA, volume 4867 of lecture notes in computer science* (pp. 188–202). New York: Springer.
35. Vaudenay, S., & Vuagnoux, M. (2007). Passive-only key recovery attacks on RC4. In C. M. Adams, A. Miri, & M. J. Wiener (Eds.), *Selected areas in cryptography, volume 4876 of lecture notes in computer science* (pp. 344–359). New York: Springer.
36. Tews, E., & Beck, M. (2009). Practical attacks against WEP and WPA. In D. A. Basin, S. Capkun, & W. Lee (Eds.), *WISEC* (pp. 79–86). New york: ACM.
37. Sepehrdad, P. (2012). *Statistical and algebraic cryptanalysis of lightweight and ultra-lightweight symmetric primitives*. Ph.D. thesis No. 5415, École Polytechnique Fédérale de Lausanne (EPFL). http://lasecwww.epfl.ch/∼sepehrdad/Pouyan_Sepehrdad_PhD_Thesis.pdf.
38. Sepehrdad, P., Vaudenay, S., & Vuagnoux, M. (2010). Discovery and exploitation of new biases in RC4. In A. Biryukov, G. Gong, & D. R. Stinson (Eds.), *Selected areas in cryptography, volume 6544 of lecture notes in computer science* (pp. 74–91). New York: Springer.

39. Sepehrdad, P., Vaudenay, S., & Vuagnoux, M. (2011). Statistical attack on RC4—Distinguishing WPA. In K. G. Paterson (Ed.), *EUROCRYPT, volume 6632 of lecture notes in computer science* (pp. 343–363). New York: Springer.
40. Sepehrdad, P., Sušil, P., Vaudenay, S., & Vuagnoux, M. (2013). Smashing WEP in a passive attack. In S. Morial (Ed.), *Internaional Workshop on Fast Software Encryption* (pp. 155–178). Berlin: Springer.
41. Knudsen, L. R., Meier, W., Preneel, B., Rijmen, V., & Verdoolaege, S. (1998). Analysis methods for (alleged) RC4. In K. Ohta & D. Pei (Eds.), *ASIACRYPT, volume 1514 of lecture notes in computer science* (pp. 327–341). New York: Springer.
42. Mister, S., & Tavares, S. E. (1998). Cryptanalysis of RC4-like ciphers. In S. E. Tavares & H. Meijer (Eds.), *Selected areas in cryptography, volume 1556 of lecture notes in computer science* (pp. 131–143). New York: Springer.
43. Golic, J. D. (2000). Iterative probabilistic cryptanalysis of RC4 keystream generator. In E. Dawson, A. Clark, & C. Boyd (Eds.), *ACISP, volume 1841 of lecture notes in computer science* (pp. 220–233). New York: Springer.
44. Shiraishi, Y., Ohigashi, T., & Morii, M. (2003). An improved internal-state reconstruction method of a stream cipher RC4. In M. H. Hamza (Ed.), *Proceedings of Communication, Network, and Information security, Track 440–088, Newyork, USA, December 10–12*, (pp.440–488). Canada: ACTA press.
45. Tomasevic, V., Bojanic, S., & Nieto-Taladriz, O. (2007). Finding an internal state of RC4 stream cipher. *Information Sciences, 177*(7), 1715–1727.
46. Maximov, A., & Khovratovich, D. (2008). New state recovery attack on RC4. In D. Wagner (Ed.), *CRYPTO, volume 5157 of lecture notes in computer science* (pp. 297–316). New York: Springer.
47. Golic, J. D., & Morgari, G. (2008). Iterative probabilistic reconstruction of RC4 internal states. *IACR Cryptology ePrint Archive, 2008*, 348.
48. Gupta, S. S., Maitra, S., Paul, G., & Sarkar, S. (2011). Proof of empirical RC4 biases and new key correlations. In A. Miri & S. Vaudenay (Eds.), *Selected areas in cryptography, volume 7118 of lecture notes in computer science* (pp. 151–168). New York: Springer.
49. Gupta, S. S., Maitra, S., Paul, G., & Sarkar, S. (2014). (Non-) Random Sequences from (Non-) Random Permutations—Analysis of RC4 stream cipher. *Journal of Cryptology, 27*(1), 67–108.
50. Isobe, T., Ohigashi, T., Watanabe, Y., & Morii, M. (2013). Full plaintext recovery attack on broadcast RC4. In *Proceedings of the 20th international workshop on fast software encryption (FSE 2013)*.
51. Sarkar, S., Gupta, S. S., Paul, G., & Maitra, S. (2013). Proving TLS-attack related open biases of RC4. *IACR Cryptology ePrint Archive, 2013*, 502.
52. Jenkins Jr, R. J. (1996). *ISAAC and RC4*. http://burtleburtle.net/bob/rand/isaac.html.
53. Mantin, I., & Shamir, A. (2001). A practical attack on broadcast RC4. In M. Matsui (Ed.), *FSE, volume 2355 of lecture notes in computer science* (pp. 152–164). New York: Springer.
54. Mantin, I. (2001). *Analysis of the stream cipher RC4*. Master's thesis, The Weizmann Institute of Science, Israel. www.wisdom.weizmann.ac.il/∼itsik/RC4/RC4.html.
55. Paul, G., Maitra, S., & Srivastava, R. (2007). On non-randomness of the permutation after RC4 key scheduling. In S. Boztas & H. F. Lu (Eds.), *AAECC, volume 4851 of lecture notes in computer science* (pp. 100–109). New York: Springer.
56. Sarkar, S. (2015). Further non-randomness in RC4, RC4A and VMPC. *Cryptography and Communications, 7*(3), 317–330.
57. Maitra, S., Paul, G., & Gupta, S. S. (2011). Attack on broadcast RC4 revisited. In A. Joux (Ed.), *FSE, volume 6733 of lecture notes in computer science* (pp. 199–217). New York: Springer.
58. AlFardan, N., Bernstein, D., Paterson, K. G., Poettering, B., & Schuldt, J. C. N. (2013). On the security of RC4 in TLS. In *USENIX security symposium*. Presented at FSE 2013 as an invited talk [14] by Dan Bernstein. Full version of the research paper and relevant results are available online at http://www.isg.rhul.ac.uk/tls/.
59. Golic, J. D. (1997). Linear statistical weakness of alleged RC4 keystream generator. In W. Fumy (Ed.), *EUROCRYPT, volume 1233 of lecture notes in computer science* (pp. 226–238). New York: Springer.
60. Fluhrer, S. R., & McGrew, D. A. (2000). Statistical analysis of the alleged RC4 keystream generator. In B. Schneier (Ed.), *FSE, volume 1978 of lecture notes in computer science* (pp. 19–30). New York: Springer.
61. Mantin, I. (2005). Predicting and distinguishing attacks on RC4 keystream generator. In R. Cramer (Ed.), *EUROCRYPT, volume 3494 of lecture notes in computer science* (pp. 491–506). New York: Springer.
62. Basu, Riddhipratim, Ganguly, Shirshendu, Maitra, Subhamoy, & Paul, Goutam. (2008). A complete characterization of the evolution of RC4 pseudo random generation algorithm. *Journal of Mathematical Cryptology, 2*(3), 257–289.

63. Jindal, P., & Singh, B. (2015). A survey on RC4 stream cipher. *Journal of Computer Network and Information Security, 2015*(7), 37–45.
64. Gong, G., Gupta, K. C., Hell, M., & Nawaz, Y. (2005). Towards a general RC4-like keystream generator. In D. Feng, D. Lin, M. Yung (Eds.), *Information security and cryptology* (pp. 162–174). Springer: Berlin.
65. Orumiehchiha, M. A., Pieprzyk, J., Shakour, E., & Steinfeld, R. (2013). Cryptanalysis of RC4 (n, m) Stream Cipher. In *Proceedings of the 6th international conference on security of information and networks*, (pp. 165–172). ACM.
66. Xie, J., & Pan, X. (2010). An improved RC4 stream cipher. In *2010 International conference on computer application and system modeling (ICCASM),* (Vol. 7, pp. V7–156). IEEE.
67. Paul, G., Maitra, S., & Chattopadhyay, A. (2013). Quad-RC4: Merging Four RC4 States towards a 32-bit stream cipher. *IACR Cryptology ePrint Archive, 2013*, 572.
68. Kherad, F. J., Naji, H. R., Malakooti, M. V., & Haghighat, P. (2010). A new symmetric cryptography algorithm to secure e-commerce transactions. In *2010 International conference on financial theory and engineering (ICFTE),* (pp. 234–237). IEEE.
69. Weerasinghe, T. D. B. (2014). An effective RC4 stream cipher. *IACR Cryptology ePrint Archive, 2014*, 171.
70. Jindal, P., & Singh, B. (2014). Performance analysis of modified RC4 encryption algorithm. In *Recent advances and innovations in engineering (ICRAIE),* (pp. 1–5). IEEE.
71. Lv, J., Zhang, B., & Lin, D. (2013). Distinguishing attacks on RC4 and a new improvement of the cipher. *IACR Cryptology ePrint Archive, 2013*, 176.
72. Khine, L. L. (2009). A new variant of RC4 stream cipher. *World Academy of Science, Engineering and Technology, 50*, 958–961.
73. Naik, K., & Wei, D. S. (2001). Software implementation strategies for power-conscious systems. *Mobile Networks and Applications, 6*(3), 291–305.
74. Farkas, K., Wellnitz, O., Dick, M., Gu, X., Busse, M., Effelsberg, W., et al. (2006). Realtime service provisioning for mobile and wireless networks. *Computer Communications, 29*(5), 540–550.
75. Jindal, P., & Singh, B. (2015). Experimental study to analyze the security performance in wireless LANs. *Wireless Personal Communications, 83*(3), 2085–2131.

Poonam Jindal received B.E. (ECE) from P.T.U. Punjab, M.E. (ECE) from Thapar University, Patiala and Ph.D. from NIT Kurukshetra. She is with ECE department of NIT Kurukshetra (India) working as Assistant Professor having 11 years of teaching experience. She has 25 research publications in international/national journals and conferences to her credit. Her research interests include wireless network security and mobile communication. She is member of IEEE.

Prof. Brahmjit Singh received B.E. degree from MNIT Jaipur, M.E. from IIT Roorkee and Ph.D. from GGS Indraprastha University, Delhi. He is with ECE department of NIT Kurukshetra (India), working as Professor with more than 27 years of teaching and research experience. He has held several administrative and academic positions in NIT Kurukshetra. These include Chairman ECE Department, Chairman Computer Engineering Department, Professor in-Charge Centre of Computing and Networking, and Member Planning and Development Board. He has published 140 research papers in international/national journals and conferences in the area of wireless communications, sensor networks, wireless network security and cognitive radios. He has been awarded The Best Research Paper Award on behalf of 'The Institution of Engineers (India)'. He is the member of IEEE, Life member of IETE, and Life Member of ISTE.