RC4D: A New Development of RC4 Encryption Algorithm

Rawan Alsharida[1], Maytham Hammood[1(✉)], Mohamed A. Ahmed[1], Barzan Thamer[1], and Mohanaad Shakir[2]

[1] Department of Computer Science, College of Computer Science and Mathematics, University of Tikrit, Tikrit, Iraq
{rawan-adel,maythamhammood}@tu.edu.iq
mohamed.aktham3@gmail.com, sdd3512@gmail.com
[2] College of Business, University of Buraimi, Al-Buraimi, Oman
mohanaad.t@uob.edu.om

Abstract. Cryptography is one of the essential methods for securing the information. In cryptography, there are many encryption algorithms; some of them strong where the others are broken. RC4 stream cipher one of the most common algorithms that are characterized by its speed in implementation does not need large storage space and has less complexity, but there are weaknesses in its output. Numerous researches work on the RC4 stream cipher to boost the security of it, to be strong enough. However, the biases in the output are still in most of the enhancement. The researchers claim that its swap function is responsible for those biases. They recommended to ignore some initial bytes from the keystream output, to dispose of this before de facto encryption begins. This paper present new development over the RC4 algorithm (RC4D) via amendment in the first and second parts of the algorithm. In the first part, it increases the use of the key operations to obtain more considerable randomness, while adding one more random variable and use the Xor function in the second part. Thus, the experimental result of the NIST statistical tests and the distant-equalities statistical analysis shows the RC4D more robust than the original RC4.

Keywords: RC4 · Stream cipher · Cryptography

1 Introduction

Encryption is vital nowadays for keeping our information confidential. We need to maintain our information through it convert into unreadable in the public channel via encryption [1]. Encryption aims to protect information from any attempt to reveal confidential data by any unauthorized third party, and the data is protected through the use of one of the algorithms for encryption. Besides, encryption in information security means encrypting data and making it incomprehensible for protection. Data can only be processed with an encryption key

B. Ghita and S. Shiaeles (Eds.): INC 2020, LNNS 180, pp. 19–30, 2021.
https://doi.org/10.1007/978-3-030-64758-2_2

that converts the regular written form into an encrypted format [2]. There are two primary types of encryption algorithms which are symmetrical and asymmetric. In the asymmetric key type use two key one of them for encryption and the other for decryption. Where in the symmetric key just use one key for encryption and decryption and key should be random. The randomness is significant, and it is challenging to be expected. Here, the strong point of this type, which is meant by randomness according to the definition, and there are two types of randomness; True Random Number Generators(TRNGs) and Pseudo Random Number Generators (PRNGs) [3]. However, the type (TRNGs) that generates randomness using the unpredictable physical process noise likes thermal noise or jitter. This type is not suitable for encryption because the source of randomness is non-deterministic for that reason the same chain cannot be regenerated but it is good to generate a random keys for encryption. Where the second type (PRNGs) generates a certain sequence of numbers by using a specific function and mathematical algorithms to generate a pseudo-random series in a deterministic way that will appear random, however, this type is used in encryption because it is fast, and can regenerate the same sequence by using the same input string as a seed. The seed must be random and long enough to increase the randomness of the output sequence. RC4 is an algorithm consisting of a simple structure that has been proven after being subjected to necessary testing and analysis by researchers and has proven to be powerful enough on various platforms. The fundamental of this algorithm is one internal state table of size N (which mostly 256), which ultimately functions as the S-box for performing the primary mixing in bytes.

The RC4 algorithm was initially designed in 1987 by Ronald Rivest and was kept as a secret until it was disclosed onto the network in 1994 [4]. This Stream Cipher algorithm, which uses the symmetric key, is an algorithm that is known by its speed, and when comparing it with the DES algorithm, it is the best in terms of speed and more straightforward in terms of complexity. In addition, it is used in various protocols, and multiple applications such as Wi-Fi protected access (WPs) and Transport Layer Security (TLS) for web browsing, emailing, and instant messaging. Many encryption operations used in wireless rely on symmetric encryption and are used in Oracle, SQL, protocol (SSL), Wired equivalency privacy protocol. In this research, the researchers design an encryption algorithm that is characterized by its speed in implementation does not need large storage space, less complicated, and is more reliable in protecting information or data. It also has a higher degree of security for devices with limited resources, and there is no bias for the first bits as in the RC4 algorithm. The remainder of the paper is ordered as comes: Literature Review in Sect. 2. Section 3 is RC4 explanation where Sect. 4 presents some weaknesses of RC4. The presentation of the modified RC4 algorithm is in Sect. 5. Performance evaluation is in Sect. 6. The implementation in Sect. 7 then ended by the conclusion in Sect. 8.

2 Literature Review

Many researchers in the field of data security found weaknesses in the security of the RC4, while many researchers tried to develop the RC4 algorithm to bypass these vulnerabilities. However, an attack on this algorithm was provided by Flunrer et al. [5], and the main goal is to eliminate the interconnection between outputs.

Thus, the creation of several modified algorithms was developed for the original RC4 algorithm and on that disability, but these algorithms that improve the original algorithm affected the speed of implementation. On the other hand, others tried to improve the speed of the algorithm but caused a reduction in randomness [6–18]. Nishith et al. developed the RC4 algorithm order to enhance the security of the RC4 by replacing along these variable lines above it to encode an element. The time it takes to encode and decode using the proposed algorithm is barely a little more substantial than the original algorithm [7]. Hammood et al. [8] introduced a random RC4 algorithm (RRC4) to pass the overcome the weak scheduling of keys in the RC4 algorithm. This algorithm enhanced the confidentiality of the encrypted text and proved more random than the original RC4 algorithm. Also, the coding and decoding time for the improved algorithm was the same time in the original algorithm. Ali et al. [9] presented an advanced algorithm based on the original RC4 algorithm, where this algorithm led to an increase in the randomness generated by using mathematical operations, which are Factorial and some other variables. In this case, the security of the algorithm was enhanced, but in contrast, during the comparison of the improved algorithm with the original algorithm, the results showed there is a clear increase in implementation time and the storage space, where the reason for this increase is the Factorial process. Variably Modified Permutation Composition (VMPC) was suggested by Zoltak's [10], which includes many changes in the initialization and the processing rounds as well as in the output generating. It was contagious to be effective in the application and solution of vulnerability in the KSA of RC4 which was described by Fluhrer et al. [5] Pseudo-Random Generation Algorithm (PRGA) structure of VMPC is sophisticated more than the RC4 which makes it higher resistant to attack. Paul and Preneel [11] through their research, introduced a new algorithm, an improved RC4A algorithm, and as a reinforcement over the RC4 algorithm, after discovering an unusual statistical weakness in generations of the first two key bytes for RC4. They said that the number of outputs demanded to distinguish an RC4 random output was 225. RC4A is powerful against most of the weaknesses in RC4, especially the weaknesses of distribution at the first two bytes output. Nevertheless, one year later Maximov introduced a distinct attack on VMPC and RC4A together. Thus distinct attack could distinguish ciphertext from a truly random number [12]. Pu and Chung, [13] presented a group key update method to improve the randomness of RC4 thus increase the security against the attacks by using the

same length of the key and initialization vector. The experimental results show this method uses minimum resource from constrained devices such as wireless sensors. The results confirm that the new produced key and the update of the key are random. Also, the randomness will improve during the abundance of sensor nodes is increased.

Hammood et al. presented [14] new improvement over RC4 by using two-state tables to rise its randomness, thus reduce the correlation between the output and the key but also the time of encryption is increased. Yao et al. [15], through his research, contributed analysis and improvement of the safety of the RC4 algorithm via applying a public key with RC4. Still, it drove to an increment in the system size and execution time. Three algorithms presented in [16] for improving RC4 each one has different properties but RC4-2S+ with two state tables to generate four key bytes in every round are the best one and more randomness without affecting its speed, but it has high bias. Suman et al [17] analysed different variants of RC4 to find out the benefit of dropping initial bytes. Also, they produced multiple S-boxes by various logics and a unique key to increasing robust of RC4 but the speed of RC4 increased. Zelenoritskaya et al. [18] presented an improvement of the RC4, by using parallel LFSR with stochastic transformation boxes. Thus, the complexity of the relationship between the key and the initial state increased. However, the using of LFSR led to an increase in the time of execution.

3 Description of RC4

Several Stream Cipher algorithms rely on the use of Linear Feedback Shift Registers (LFSRs), particularly in physical components, but when the RC4 algorithm was de-signed (LFSRs) were not used. RC4 algorithm involves two parts to produce the first component key. It is a Key Scheduling Algorithm (KSA), as shown in Algorithm 1. The second component is the Pseudo-Random Generation Algorithm (PRGA), as shown in Algorithm 2, which is executed consecutively [4].

RC4 algorithm has a variable key size domain between 8 to 256 bytes to initialize 256 bytes in the initial state array by the number from 0 to 255. The KSA algorithm produces flipping or initial switching by mixing the flipping or the corresponding switch by using the key, and the result of this flipping is considered an entrance to the part PRGA. This step creates the final key RC4 starts with switching, and the secret key is used to produce random swapping. The next phase is the PRGA algorithm that generates bytes of the keystream, which performs the XOR process with the plaintext to obtain the ciphertext. The concept of the RC4 algorithm is to make the flipping or switching of elements by exchanging them with each other for higher randomization.

Algorithm 1. RC4 - KSA algorithm

Set $N \leftarrow 256$
Set $i \leftarrow 0$
while (i<N) **do**
  $S[i] \leftarrow i$
  $i \leftarrow i + 1$
end while
Set $i \leftarrow 0$
Set $j \leftarrow 0$
while (i<N) **do**
  $j \leftarrow (j + S[i] + K[i \ mod \ L])mod \ N$
  Swap ($S[i]$, $S[j]$)
end while
return S

Algorithm 2. RC4 - PRGA algorithm

Set $i \leftarrow 0$
Set $j \leftarrow 0$
while *generate key-stream* **do**
  $i \leftarrow (i + 1)mod \ N$
  $j \leftarrow (j + S[i])mod \ N$
  Swap ($S[i]$, $S[j]$)
  $Output \leftarrow S[(S[i] + S[j])mod \ N]$
  $Ciphertext \leftarrow Output \oplus Plaintext$
  return *Ciphertext*
end while

4 The Cryptanalysis of RC4

Cryptanalysis is used to violate cryptographic security systems by breaking encrypted messages without the known cryptographic key. It mainly focuses on identifying the non-random procedure. There are many weaknesses in the RC4 algorithm a set of vulnerabilities that can be solved, and the other group is considered a threat by attackers to exploit. One of the weaknesses in the case of initialization is the statistical bias that occurs in word distribution in the first outlet. The problem of the algorithm in the first component (KSA) is simple but random. There is a relationship between some byte units of the secret key, so the analysis of these bytes makes it possible to attack RC4. The key flow that starts at the beginning of the algorithm is a one-time swaps S (identical to the value of (i) the pointer indicating the entry) for low values, and S [j] = j is possible during initialization [19]. Roos [20] found another weakness in RC4, which is a correlation among the values of the first table and the keystream. The primary reason is the status table that started in the sequence (0, 1, 2, ...,

255) and at least one of the 256 temporary keys. The first byte that was created from the key is highly correlated with a few first bytes of PRGA output; thus, to solve this issue, it has been advised to discard the initial byte of PRGA output. The main objective of the adversary is to obtain the original key, internal state, or output key to access the plaintext. Since prior studies of RC4, there are some weaknesses in it, such as biased bytes, and key recovery where various attacks focus on obtaining the private key for the internal state [21]. Tomašević et al. [22] displayed a cryptanalytic attack that utilizes the tree design of this cipher and offers an abstraction of prevailing conditions for the information of its state table. To obtain the initial state will using the hill-climbing strategy for the search in the tree of general conditions. The complication of this attack is weaker than that of an exhaustive search. The successful attack on RC4 utilized the existence of biases in the RC4 keystream to make plaintext recovery attacks against TLS-RC4 [23]. Hammood et al. [24] evaluated the likelihood of the generating ciphertext for each pair of byte values for each 256-byte round than found new four biases.

5 Modified RC4 Algorithm

Due to the existence of many weaknesses in the RC4 algorithm, as mentioned earlier, in this research, the RC4 algorithm was improved through modification in the first and second parts. In the first part, as shown in Algorithm 3, RC4D-KSA was modi-fied through an increase using the key and generates new random variable to obtain more randomness, while in the second part, as shown in Algorithm 4; RC4D -PRGA the algorithm was modified by generate new random variable and XORed it with the random j then find $S[S[j] \oplus S[T]]$. Thus, the randomness will increase. The modifi-cation in the RC4D- KSA does not take much time, because they only happen 265 times.

Algorithm 3. RC4D - KSA algorithm

Set $N \leftarrow 256$
Set $i \leftarrow 0$
while (i<N) **do**
  $S[i] \leftarrow i$
  $i \leftarrow i + 1$
end while
Set $i \leftarrow 0$
Set $j \leftarrow 0$
while (i<N) **do**
  $j \leftarrow (j + S[(i + K[i\ modL])modN] + K[i\ modL])\ modN$
  Swap ($S[i]$, $S[j]$)
end while
return S

Algorithm 4. RC4D - PRGA algorithm

Set $i \leftarrow 0$
Set $j \leftarrow 0$
Set $D \leftarrow 0$
while *generate key-stream* **do**
  $i \leftarrow (i+1) mod\ N$
  $j \leftarrow (j + S[i]) mod\ N$
  Swap $(S[i]\ ,\ S[j])$
  $Output \leftarrow S[(S[i] + S[S[j] \oplus S[D]]) mod\ N]$
  Set $D \leftarrow Output$
  $Ciphertext \leftarrow Output \oplus Plaintext$
  return *Ciphertext*
end while

6 Performance Evaluation

There is a set of different criteria for measuring the level of safety and performance of the specific encryption algorithm. At this time, two measurements were used. The first test was distant-equalities statistical test [25] and the second test a random statistical Suite test [27] created by the NIST (National Institute of Standards and Technology). In addition, evaluated the time of the performance of the proposed algorithm.

6.1 Distant-Equalities Statistical Test

As mentioned, there are weaknesses in Stream Cipher algorithms in the process of generating keys; there is bias in randomization. This test works to measure this bias by generating a large number of keys (N) and then measures how similar these keys are between them [25,26]. Simply this test each key with the eight keys followed it's by a matrix consisting of 8 different events: Event Z_i for $i \in 1, 2, 3, 4, 5, 6, 7, 8$ The first event Z_1 test the first key with the second key, Event two Z_2 test the first key with the third key, Event three Z_3 test the first key with the fourth key, Event four Z_4 test the first key with the fifth key, Event five Z_5 test the first key with the sixth key, Event six Z_6 test the first key with the seventh key, Event seven Z_7 test the first key with the eighth key, Event eight Z_8 test the first key with the ninth key. Note t : (start test) begins from K1 to Kn. Z_i++ in every t for i =1,2,3,4,5,6,7,8. The probability of each event in the output is p = 1/N, where N is the keyword size of the algorithm. However, the expected number of occurrences for every event of n samples is present in Eq. 1. [25]

$$E = np \tag{1}$$

In addition, the standard deviation of the number of occurrences is given by the binomial distribution as shown in Eq. 2 [25].

Table 1. Distant-equalities statistical test results for RC4.

Samples	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Event 8
2^{25}	25.9838	−65.9029	26.6885	−24.9691	13.7112	0.489631	4.57946	2.93726

$$\sigma = \sqrt{(np(1-p))} \tag{2}$$

The test calculations the numbers of occurrences for the Events from one to eight then compares them with the results of model E = np utilize σ as the measure of deviation as shown in Eq. 3. [25]

$$d = (f - E)/\sigma \tag{3}$$

Where d is the test calculates, and f represents the measured number, which is standard statistical practice. Table 1 represents the amounts of standard deviations that measured numbers of appearing for Events from Event1 to Event8 were different compere with their estimated values (d = (f − E)/σ) for different word sizes (N = 8) operates. We found a clear bias between the Event1 and the Event2 that the Event2 approximates 3σ of the Event1 almost three times the first event while otherwise, the bias may not be in the Event6 and Event7.

Table 2. Distant-equalities statistical test results for RC4D.

Samples	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Event 8
2^{25}	1.86613	−2.27851	−1.36554	−0.271437	2.04778	1.33317	0.979784	−0.887913

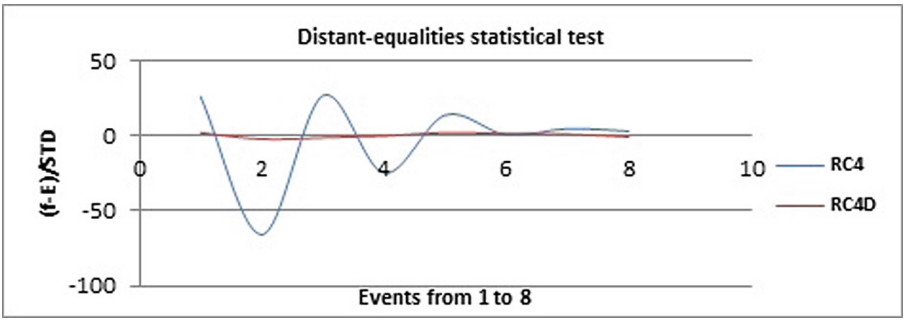

Fig. 1. Distant-equalities statistical test results for RC4 and RC4D.

It was observed that when applied the test over original RC4, the bias is very clear, in most the events spatially in the first five events as shown in Table 1. In

contrast, the results of RC4D showed it's very good there is no bias when use the same amount of key as shown in Table 2. The results indicate the randomness increases because the authors add a new random parameter. Thus, the RC4D is given better results than RC4. As shown Fig. 1 which describes the difference between RC4 and the new RC4.

6.2 Randomness Test

The generated keystream is examined by a set of NIST (National Institute of Standards and Technology) tests, which is a statistic set of random number generation tests that include 16 statistical analyses to perform randomization of the output chain of a random number or an integer random number generator [27]. PRGA tests were performed using NIST, and the probability of a valid random number generator was represented with a value of P. These tested are produced and the simplified mean of the P-values from these tests, as shown in Table 3. In the test, the P-values are compared to 0.01, the P-values are passed when they are larger than 0.01, and the series produced is random and uniformly distributed. If the tests give a value of P equal to 1, then the string is a completely random series, but if the P-values is zero, it means that the string is entirely random. Success means that the string is acceptable and has good randomness,

Table 3. NIST Tests Applied to Standard RC4 and Modified RC4 (RC4D).

Statistical Test Name	RC4		RC4D	
	p-value	Result	p-value	Result
Approximate Entropy	0.436169	Pass	0.541461	Pass
Block Frequency	0.325088	Pass	0.438262	Pass
Cumulative Sums (Forward)	0.581981	Pass	0.602363	Pass
Cumulative Sum (Reverse)	0.448262	Pass	0.638262	Pass
FFT	0.457888	Pass	0.497533	Pass
Frequency	0.598393	Pass	0.625319	Pass
Lempel-Ziv compression	1	Pass	1	Pass
Linear Complexity	0.422196	Pass	0.590875	Pass
Longest Runs	0.430753	Pass	0.571836	Pass
Non periodic Templates	0.472041	Pass	0.534455	Pass
Overlapping Template	0.476283	Pass	0.451715	Pass
Random Excursions	0.50275	Pass	0.552624	Pass
Random Excursions Variant	0.588278	Pass	0.506814	Pass
Rank	0.493697	Pass	0.536438	Pass
Runs	0.406088	Pass	0.568629	Pass
Serial	0.558872	Pass	0.450122	Pass
Universal Statistical	0.557777	Pass	0.697207	Pass

while failure indicates that it is not acceptable and not random. Some of the 16 tests accepted large sizes of output series and failed in the small size, as well as other tests accepted large and small sizes of output series. In our program, a large volume (1000000 bytes) of each secret key is created. In this paper, as shown in Table 3 the result indicates that the RC4D is better than the original RC4 in most of the tests.

7 Implementation

When implementing the algorithm using the C++ language, the values in the algorithm inputs in the initial state from 0 to 255, and the key length was 16 byte only. The time of execution in both algorithm the original RC4 and RC4D is approximately the same as shown in Table 4.

Table 4. Execution time of RC4 and RC4D.

Key Size (kilobyte)	RC4 Time (m.s)	RC4D Time (m.s)
100,000	12.488	12.518
1,000,000	117.849	118.315

8 Conclusions

In this paper, the author presents an improvement over the RC4 encryption algorithm. It is also more reliable in protecting information or data and has a higher degree of security for devices with limited resources; also, there is no bias for the first bits as in the original algorithm With the increasing of modern technology nowadays the challenge is how to secure the data and confidential information against the weak souls, which are electronic armies and malicious intentions have piqued these companies. For fear of theft of these data and information has been encrypted, this data and information using different encryption algorithms, but these algorithms contain vulnerabilities exploited by hackers to steal data and information. Thus, to increase security and reduce potential attacks should be used robust encryption algorithms. In this research, we presented improvement over RC4 and test the result by NIST statistical test and a distance statistical test. These tests help to detect the weaknesses of the stream cipher algorithms. The test is distinguished from the previous tests at its speed, as well as it is characterized by giving results of high accuracy and efficiency, so it is useful in the process of improving the cipher algorithms. These characteristics of RC4D algorithm make it a better nominee for effective applications as compared to the original RC4 Algorithm.

References

1. Rueppel, R.A.: Analysis and design of stream ciphers. Springer, New York (2012). https://doi.org/10.1007/978-3-642-82865-2
2. Lamba, C.S.: Design and analysis of stream cipher for network security. In: Second International Conference on Communication Software and Networks, Singapore, pp. 562-567 (2010). https://doi.org/10.1109/ICCSN.2010.113
3. Thomas, D.B., Luk, W.: High quality uniform random number generation using LUT optimised state-transition matrices. J. VLSI Signal Process **47**, 77–92 (2007). https://doi.org/10.1007/s11265-006-0014-9
4. Rivest, R.L.: The RC4 encryption algorithm, RSA Data Security Inc., **12**9-2, March 1992. This document has not been made public
5. Fluhrer, S., Mantin I., Shamir, A.: Weaknesses in the key scheduling algorithm of RC4. In: Proceedings of Annual Workshop on Selected Areas in Cryptography, Springer, Heidelberg, vol. 2259, pp. 1-24 (2001). https://doi.org/10.1007/3-540-45537-X_1
6. Xie, J., Pan, X.: An improved RC4 stream cipher. In: International Conference on Computer Application and System Modeling (ICCASM), IEEE vol. 7, pp.156-159 (2010). https://doi.org/10.1109/ICCASM.2010.5620800
7. Sinha, N., Chawda, M., Bhamidipati, K.: Enhancing security of improved RC4 stream cipher by converting into product cipher. Int. J. Comput. Appl. (0975 - 8887) vol. 94 - No .18, pp. 17–21, May 2014. https://doi.org/10.5120/16459-6132
8. Hammood, M.M., Yoshigoe, K., Sagheer, A.M.: RC4 stream cipher with a random initial state. In: Proceedings in Information Technology, Springer, Dordrecht, p. 407 (2013). https://doi.org/10.1007/978-94-007-6996-0_42
9. Sagheer, M.A., Searan, S.M., Alsharida, R.A.: Modification of RC4 algorithm to increase its security by using mathematical operations, J. Software Eng. Intell. Syst. **1**(2), ISSN 2518-8739 (2016). http://www.jseis.org/Volumes/Vol1/V1N2-1.pdf
10. Zoltak, B.: VMPC One way Function and Stream Cipher, Fast Software Encrypt. FSE, LNCS, 3017, pp. 210–225. Springer, New York (2004)
11. Paul, S. Preneel, B.: A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In: Fast Software Encrypt, FSE, LNCS 3017. New York: Springer, Heidelberg, pp. 245–259 (2004). https://doi.org/10.1007/978-3-540-25937-4_16
12. Maximov, A.: Two linear distinguishing attacks on VMPC and RC4A and weakness of the RC4 family of stream ciphers. In: Fast Software Encryption, FSE, Vol 3557, pp. 342-358, Springer, Cham (2005). https://doi.org/10.1007/11502760_23
13. Pu, C., Chung, W.C.: Group key update method for improving RC4 stream cipher in wireless sensor networks. In: International Conference on Convergence Information Technology, Gyeongju, pp. 1366-1371 (2007)
14. Hammood, M.M. Yoshigoe, K., Sagheer, A.M.: RC4-2S: RC4 stream cipher with two state tables. In: Information Technology Convergence, pp. 13–20. Springer, Netherlands (2013). https://doi.org/10.1007/978-94-007-6996-0_2
15. Yao, Y., Chong, J., Xingwei, W.: Enhancing RC4 algorithm for WLAN WEP protocol. In: Control and Decision Conference (CCDC), pp. 3623–3627, IEEE (2010). https://doi.org/10.1109/CCDC.2010.5498536
16. Hammood, M.M., Yoshigoe, K., Sagheer, A.M.: Enhancing security and speed of RC4. Int. J. Comput. Network Technol. **3**(02) (2015). https://doi.org/10.12785/ijcnt/030201

17. Das, S., Ghosh, R., Pal, R.K.: An approach of refining RC4 with performance analysis on new variants. Sādhanā **44**(11), 223 (2019). https://doi.org/10.1007/s12046-019-1209-7
18. Zelenoritskaya, A.V., Ivanov, M.A., Salikov, E.A.: Possible modifications of RC4 stream cipher. In: Advanced Technologies in Robotics and Intelligent Systems, pp. 335–341. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-33491-8-40
19. Mister, S. Tavares, S.E.: Cryptanalysis of RC4-like Ciphers. In: International Workshop on Selected Areas in Cryptography, Springer, Cham (1998). https://doi.org/10.1007/3-540-48892-8_11
20. Roos, A.: A class of weak keys in the RC4 stream cipher, September 1995. http://agreg.dnsalias.org/Luminy/WeakKeys-report.pdf
21. Jindal, P., Singh, B.: Performance analysis of modified RC4 encryption algorithm. In: Recent Advances and Innovations in Engineering (ICRAIE), 2014. IEEE (2014). https://doi.org/10.1109/ICRAIE.2014.6909247
22. Tomašević, V., Bojanić, S., Nieto-Taladriz, O.: Finding an internal state of RC4 stream cipher. Inf. Sci. **177**(7), 1715–1727 (2007)
23. AlFardan, N., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.: On the security of RC4 in TLS. In: Presented as Part of the 22nd USENIX Security Symposium, pp. 305-320 (2013)
24. Hammood, M.M., Yoshigoe, K.: Previously overlooked bias signatures for RC4, 4th Inter-national Symposium on Digital Forensic and Security (ISDFS). Little Rock, AR **2016**, 101–106 (2016). https://doi.org/10.1109/ISDFS.2016.7473526
25. Zoltak, B.: Statistical weaknesses in 20 RC4-like algorithms and (probably) the simplest algorithm free from these weaknesses - VMPC-R, IACR Cryptology ePrint Archive, IJCSN, 2014. https://eprint.iacr.org/2014/315.pdf
26. Zoltak, B.: Statistical weakness in Spritz against VMPC-R: in search for the RC4 replace-ment. IACR Cryptology ePrint Archive, 985, (2014). https://eprint.iacr.org/2014/985.pdf
27. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Van-gel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST special publication 800–22. Gaithersburg, National institute of standards and technology (NIST) (2001). https://apps.dtic.mil/dtic/tr/fulltext/u2/a393366.pdf