International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 7, No.4 (July-2018)

http://dx.doi.org/10.12785/ijcds/070404

Developing RC4 Algorithm Using S-Box of Advanced Encryption Standard Cipher

Ali M. Sagheer [1], Sura M. Searan[1] and Salih S. Salih[1]

[1]Department of Information Technology, University of Anbar, Anbar, Iraq

Received 5 Dec. 2017, Revised 21 Jan. 2018, Accepted 15 Feb. 2018, Published 1 July 2018

Abstract: RC4 stream cipher is one of the most significant symmetric cryptosystems, it is simple and used in many commercial products. RC4 uses dynamic permutations and avoids using Linear Feed Back Shift Register (LFSR). It has many weaknesses, such as the tendency in the generated key stream that some key bytes are biased toward different values. This paper presents a new algorithm using S-box of Advanced Encryption Standard (AES) to solve the correlation between public known outputs of the internal state. The state table is filled from S-box values and additional swapping operations are used. The analysis of the proposed algorithm over variable key length produces key byte streams that have no single and double bias. This paper obtains a new algorithm that combines the efficiency of the RC4 and robustness of AES. The results show that the sequences that are generated by the developed RC4 are more random than the sequence that was generated by the RC4. Also, the developed algorithm demands little time more than RC4 execution time. Additionally, the developed algorithm is robust against most attacks, such as distinguishing attack and can be used in different protocols such as Secure Sockets Layer (SSL) Protocol, Oracle Secure SQL, and Wired Equivalent Privacy (WEP) Protocol.

Keywords: RC4, Stream Cipher, S-box, Key Scheduling Algorithm (KSA), Pseudo Random Generation Algorithm (PRGA), Advanced Encryption Standard (AES), Single Bias, Double Bias.

1. INTRODUCTION

Encryption is a process that transforms plaintext into cipher text. It is basically used to ensure confidentiality. Organizations and companies are encrypting their data before transmitting in order to ensure secure data transmission in a public channel. Cryptographic algorithms are designed to be characterized by high speed of implementation, lower size, less complexity, and larger degree of security. Conventional cryptographic algorithms are complex and take a higher amount of energy when they are used by resource constrained devices in order to provide secure communication. Indeed, public key algorithms are still not appropriate in tracer networks for many reasons, such as finite storage and higher usage of energy. Therefore, security systems should be based on a symmetric key cryptography, especially in the systems that have limited hardware resources [1]. The strength of a stream cipher is the random key stream that assures secure computation of the cipher. The cryptanalysis of stream ciphers is essentially focused on identifying non-random proceeding; till date, the analysis of stream ciphers has been employed to identify the happening of non-random proceedings [2]. The same algorithm is used for encryption and decryption; the plaintext stream is XOR-ed with the generated series of the random key generator. RC4 algorithm is used in many wireless network systems and protocols [3]. It is used in SSL protocol, Oracle Secure SQL, WEP Protocol; it is also used to protect wireless networks as part of WPA protocol and to protect the internet traffic as part of the TLS (Transport Layer Security) protocol [4]. There are many attacks presented to analysis RC4 by [5]. RC4 is analyzed by different cryptanalysis according to RC4 different weaknesses [6]. The modern researches proved that you can practically utilize single and double byte biases for RC4 to acquire any part of the Internet traffic, depending on TLS (Transport Layer Security) with RC4 option. The objective of this suggestion is to develop RC4 algorithm and analyzing the developed algorithm and shows that this algorithm is free from single and double bias while RC4 shows the bias that proved in the previous researches.

E-mail: ali.m.sagheer@gmail.com, surasms917@gmail.com, Salihsami90@gmail.com

2. Related Works

Many researchers work on analyzing the RC4 algorithm based on its weakness and suggest different algorithms. Prasithsangaree and Krishnamurthy (2003)[7] worked on analyzing RC4 and AES algorithms based on energy consumption. They determined that RC4 is more suitable for large packets, while AES symmetric algorithm is more suitable for small packets; further, RC4 is faster than AES. Maitra and Paul (2008) [8] worked on analyzing RC4 based on weakness in biases and proposed additional layers over the key scheduling algorithm and pseudorandom generation algorithm. In the same year, they determined the bias that can be perceived in S [S [y]] based on this form of permutation bias after the Key Scheduling Algorithm (KSA); a total work is presented to demonstrate that many key stream output bytes of RC4 are highly biased towards several linear collections of private key bytes (Maitra & Paul, 2008a) [5]. Al-Fardan et al. (2013) [2] determined the security of RC4 in Transport Layer Security (TLS) and Wi-Fi Protected Access (WPA) and applied single and double byte bias attack on RC4 and could retrieve some plain text bytes. In the same year, Hammood, Yoshigoe, and Sagheer (2013) [1] suggested RC4 stream cipher with two state tables (RC4-2S) as an enhancement for RC4. This enhancement solves the correlation problem between public known outputs of the internal state using permutation between (State1) and (State 2). In addition, the time period to generate the key of RC4-2S is faster than that original RC4, reduces the number of required operations in key generation. Also, Hammood, Yoshigoe, & Sagheer, 2015) [9] worked on enhancing security and speed of RC4 by proposing algorithms to enhance RC4, solve weak keys problems, and make it robust by using random initial state. The weaknesses in RC4 still represent an open challenge for developers.

3. Description of RC4 Cipher

The RC4 algorithm was proposed by Ron Rivest in 1987 and kept secret as a trade until it was leaked in 1994 [10]. It is a set of stream words of size *n*-bits [11]. RC4 starts with the permutation and uses a secret key to produce a random permutation with KSA. Based on the secret key, the next stage is Pseudo Random Generator Algorithm (PRGA) that generates key stream bytes which XORed with the original bytes of plaintext to produce the cipher text [8]. The state table is used to get pseudo-random bytes. This is done in the first phase of the algorithm [7]. The key is sometimes used as a 128-bit key. This operation is performed between key and plain text equivalent to Vernam cipher [12]. Many stream cipher algorithms use LFSR, especially in hardware architecture, but RC4 design does not. RC4 has a variable length of key that ranges between (0-255) bytes to initialize a 256-byte array in initial state (State [0] to State [255]) [1]. RC4 operated in two phases: the first consists in KSA, which initializes the internal state.

```
Algorithm 1. KSA of RC4
Input: Key
Output: State
1.    For (i = 0 to 255)
   1.1.  State[i] = i
2.    Set j = 0
3.    For (i = 0 to 255)
   3.1.  j = (j + State[i] + Key [i mod key-length]) mod 256
   3.2.  Swap (State[i], State[j])
4.    Output:  State
```

The second is a PRNG. It generates the output key stream.

```
Algorithm 2. PRGA of RC4
Inputs: State, Plaintext_i
Outputs: Key sequence (K sequence)
1.    i = 0, j = 0
2.    For (i = 0 to Plaintext length)
   2.1.  i = (i + 1) mod N
   2.2.  j = (j + State[i]) mod N
   2.3.  Swap (State[i], State[j])
   2.4.  K sequence = State [State[i] + State[j]] mod N
3.    Output:  K sequence
```

The output sequence of key K is XORed with the plaintext

$C_i = K_i \oplus \text{Plaintext}_i$ [13].

4. RC4 Weaknesses

The RC4 algorithm shows several weaknesses; some can be worked out, but others are difficult to resolve. One of these weaknesses in the initialization state is the statistical bias which occurs in distributing words of the first output. This bias makes it slight to distinguish between many short output of RC4 and random strings by analyzing their second word. This weakness is used to make effective cipher-text-only attack on this algorithm in broadcast applications, where the same plaintext is sent to multiple receivers with different keys. The unique statistical behavior is independent from the KSA and remains applicable even when the RC4 begins with a totally random permutation [14]. The slide in search effort from this attack is 25:1, but, when using linearly related session keys, the slide in effort increments to 218, that causes the weak keys [15]. Roos found weaknesses in RC4 that show a robust correlation between generated value and the first few values of the state table [16]. The main cause is the state table began in series (0, 1, 2, …., 255) and at least one out of every 256 possible keys, the first byte of the generated key, is highly correlated with a few key bytes. So, the keys allow for the precursor of the first bytes from the PRGA output [9]. The goal of the attack is to retrieve the original key, the internal state, or the output key stream to have access to the original messages. From the previous studies based on KSA and PRGA, RC4 shows the following weaknesses: biased bytes, distinguishers, key collisions, and key recovery from the state [1]. Mantin and Shamir found the major weakness of the algorithm in the second round is the probability of zero output bytes [17]. Fluhrer and McGrew found a serious weakness: anyone who knows a portion of the private key can potentially attack fully on the RC4 [18]. Maitra and Paul found a secret key by using the initial state table. Specifically, these authors generated an equation on the basis of the initial state table, selected some bytes of the secret key based on their assumption, and found out the private key by using the equation [8]. The attack aims to retrieve the main key, the internal state, or the final key stream to access to the original messages [19].

5. The Proposed Algorithm (RC4 with S-Box of AES)

This section presents a new development of the RC4 algorithm by using S-box of the AES algorithm. The idea of this proposition is taken from Rijndael algorithm. The substitution bytes of the AES is a nonlinear transformation that uses 16 bytes of S-Boxes tables, S-Box is the multiplicative inverse of a Galois field *GF* (2^8) followed by affine transformation [7]. This suggestion aims to combine the robustness and the security of the AES algorithm with the speed and the simplicity of the implementation of the RC4. More in detail, the initial state table contents are substituted with the elements of S-box to eliminate the correlation between the internal state and public known output and to reduce the weakness that is exploited by the attacks by increasing the randomness and the complexity. This algorithm starts with the initialization KSA algorithm and then the PRGA algorithm, as shown in Figure 1 below. All operations are implemented mod State length. The KSA takes a secret key *k* with a 128 *n*-bit long word in the first step; the state tables are filled by numbers from 0 to N-1 and then substituted by S-box. The input secret key is used as a state table seed. After the KSA, the PRGA performs additional swapping operations between state[i] and state[i+1], and between state[j] and state[j+1], to generate the key stream that will XORed with the plaintext to get the cipher text.

The first phase is KSA:

Algorithm 3. KSA for RC4 with S-Box of AES
Input: Secret Key
Output: State

```
1.  S-box [256] = S-box of AES algorithm
2.  For (i = 0 to N – 1)
        2.1 State[i] = S-box(i)
3.  Set j = 0
4.  For (i = 0 to N – 1)
        4.1 j = (j + State[i] + Key [i mod key-length]) mod N
        4.2 Swap (State[i], State[j])
5.  Output: State
```

The other is PRGA phase as shown below:

Algorithm 4. PRGA for RC4 with S-Box of AES
Inputs: State Table, Plaintext
Outputs: Key sequence (K), Ciphertext (C)

```
1. Initialization:
        1.1 i = 0
        1.2 j = 0
2. For (i = 0 to P_Length)
        2.1 i = (i + 1) mod N
        2.2 j = (j + S-box(j) + State[i]) mod N
        2.3 j2 = (j2 + S-box(j2) + State[i]) mod N
        2.4 Swap (State[i], State[j2])
        2.5 For (j = 0 to N – 1)
            2.5.1 Swap (State[j], State[j+1])
        2.6 K sequence = State [(State[i] + State[j] + S-box (j2 mod N) ) mod N]
        2.7 $C_i = K_i \oplus P_i$
3. Output: K sequence and $C_i$
```

The model of double RC4 with S-box of AES is shown in figure 1.

Initial with numbers from 0 to State length. Fill with chosen key.

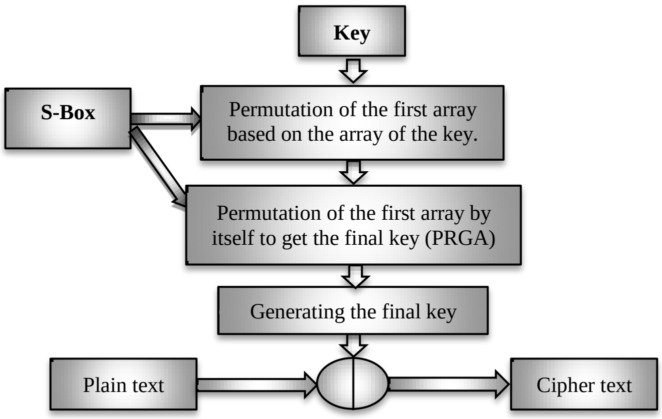
Key
S-Box
Permutation of the first array based on the array of the key.
Permutation of the first array by itself to get the final key (PRGA)
Generating the final key
Plain text
Cipher text

Figure 1. The model of developed RC4 encryption algorithm.

6. The Analysis of RC4 and Developed RC4 Algorithm Based on Single Byte Bias

Mantin and Shamir (2002) [17] were the first researchers that denoted bias in the key stream of the RC4; their result was highly accurate. Sarkar, Gupta, Paul, and Maitra (2015) [4] determined a key-length-dependent bias in the key streams of the RC4 and worked with 256-byte keys. (AlFardan, Bernstein, Paterson, Poettering, & Schuldt, 2013) [2] denoted additional biases in the key stream of RC4 that do not have theoretical observations. In this work, these researchers analyzed the RC4 and the proposed algorithm. The proposed algorithm has no bias in the key distribution bytes as determined below as a result of the use of additional operations that cause no correlation between internal state and the output sequence. Algorithm 5 is used to measure the distribution of key stream bytes.

Algorithm 5. Measuring distributions of key stream bytes

Input: Key [k_1, k_2, …., k_{16}].

Output: Key position (Kp), Key value (Kv), and the number of Frequents in each position (Kf).

```
1. For (x = 1 to $2^{21}$) Do
        1.1 i = 0
        1.2 j = 0
        1.3 Call Algorithm 1: KSA
        1.4 Call Algorithm 2: PRGA.
        1.5 Deducting new key with length = 16 from
        each generated key to be a new secret key.
2. For (i = 1 to N)
        For (j = 1 to values. Count)
        2.1 If (values [i] == value)
        2.2 Increment count by 1
        2.3 Frequents = (count / ($2^{21}$ * 16))
3. Return Kp, Kv, and Kf for each position of key stream bytes.
```

The state table is analyzed with 32 positions to reduce the search space and 2^{21} secret keys, each one with length 16, and produces 32 positions to calculate the frequent of each of the 32 values in each position. The key distribution bytes of the RC4 and the modified RC4 are determined in the following charts.

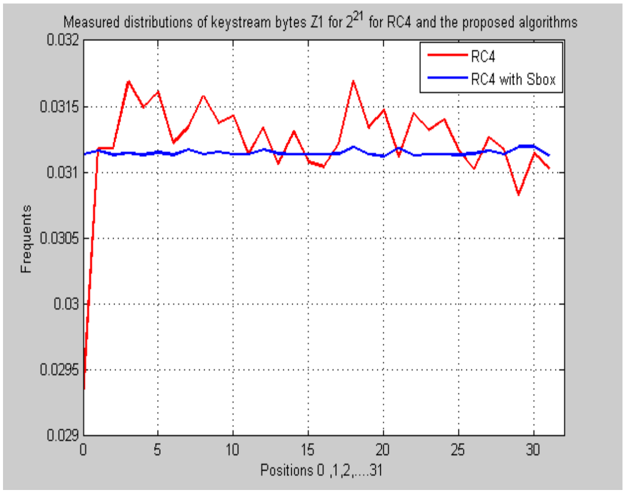

Figure 2. Key distribution bytes in the first position with 2^{21} for the RC4 and the developed RC4.

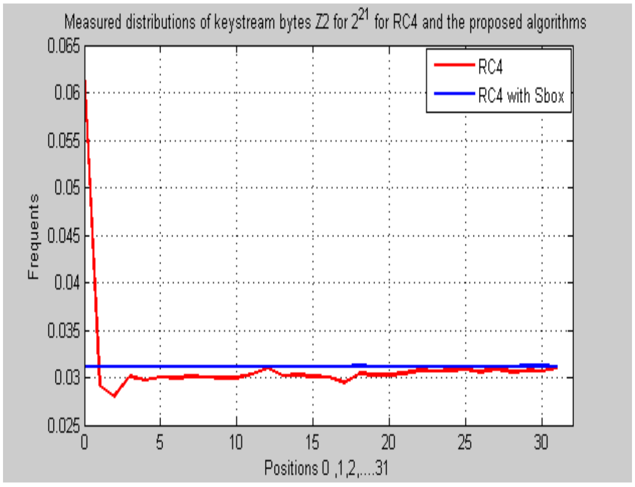

Figure 3. Key distribution bytes in the second position with 2^{21} for the RC4 and the developed RC4.

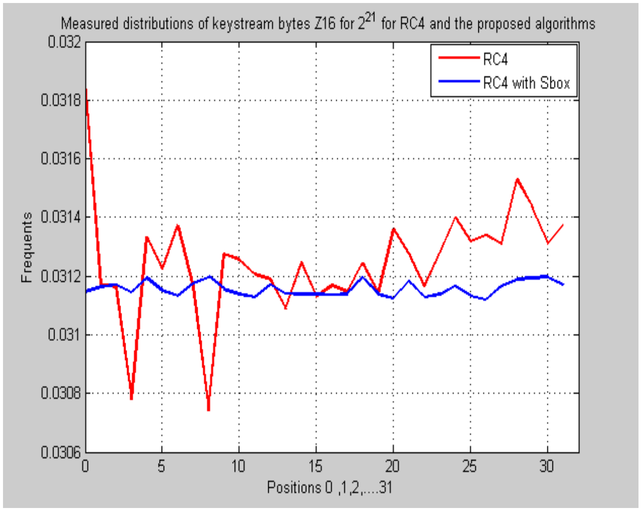

Figure 4. Key distribution bytes in the 16th position with 2^{21} for the RC4 and the developed RC4.

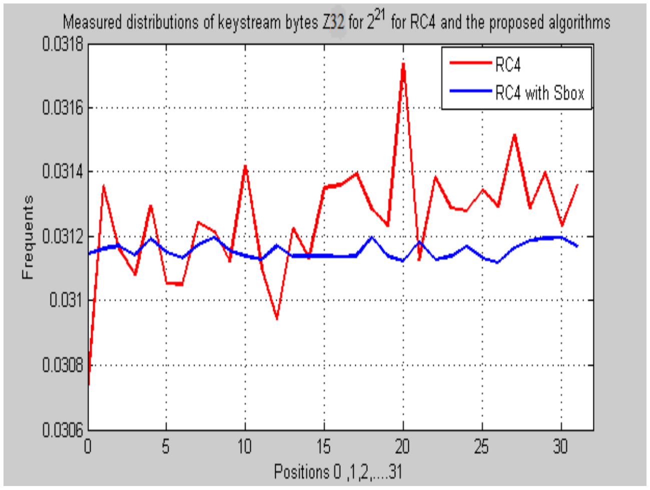

Figure 5. Key distribution bytes in the 32th position with 2^{21} for the RC4 and the developed RC4.

The distribution of key stream bytes of the RC4 algorithm has shown the same biases that are observed in literature. The experiment executed with generated key stream of 32 bytes and the number of generated keys ranging from 2^{16} to 2^{21} with independent random secret keys of 16 bytes. Expected biases started to appear for runtime beyond 2^{16} key generations, as shown in figure 2 to figure 5. They became apparent when the generated key stream increased to 2^{20}. The proposed algorithm shows that there is no bias in its key distribution bytes and the implementation time of the key stream generation is more than that required for the implementation of the RC4. The complexity and randomness in the proposed algorithm key is higher than the RC4 key bytes.

This algorithm is implemented in C#.Net programming language. Several biases were identified in literature. The RC4 successfully reproduced and proved these biases in the first 32 bytes of the key stream, while the developed RC4 has no bias in the first 32 positions of the key stream bytes.

7. The Analysis of RC4 and Developed RC4 Algorithm Based on Double Byte Bias

After explaining single-byte biases, that are significant to the cryptographic society, the attack simply can be avoided by ignoring the initial bytes. Thus, the RC4 with additional configuration can still be resistant to the single-byte bias attack. However, several authors have investigated biases beyond initial bytes and have discovered different multi-byte biases in the key stream of the RC4.

Fluhrer and McGrew (2001) [18] were the first researchers that discovered the biases in a consecutive pair of bytes (K_i, K_{i+1}) and detected long-term biases of the RC4. They discovered ten positive biases that mean their probability was higher than the desired value; besides, they detected two negative biases that mean their probability was lower than the desired value. Hammood and Yoshigoe (2016) [13] estimated the probability of the cipher for generating each pair of byte values through each 256 byte cycles and got a complete view of the distributions of every pair of byte values at the positions (i, i + 1). They replicated Fluhrer and McGrew's biases and indorse their work by Al-Fardan et al.'s (2013) studies. They found two new positive biases not mentioned by Fluhrer and McGrew (2001).

This work reproduced Fluhrer and McGrew's (2001) biases and Hammood and Yoshigoe (2016) bias with 1024 keys of 16 bytes to generate 2^{32} keystream bytes after discarding the first 1024 bytes. Each key from the 1024 keys generates 2^{32}; therefore, the whole amount of generated keys is 2^{42}. The developed RC4 using S-box did not generate any statistical bias and its output in the range only $\pm 2^4$ from the predicted occurrences. Algorithm 6 below determines the measuring of double byte bias. The main idea of this algorithm is to measure the appearance of the pair (Z_i, Z_{i+1}) in each position of the output of the RC4.

Algorithm 6. Measuring distributions of key stream bytes (K_a, K_{a+1})

Input: K [k_1, k_2, …., k_{16}]

Output: 3-Dimentions array

```
1. i = j = i1 = k = 0
2. For (x = 1 to $2^{10}$)
    2.1. Call Algorithm 2.1: KSA
    2.2. For (R= 1 to $2^{32}$)
        2.2.1. i = (i + 1) mod N
        2.2.2. j = (j + State[i]) mod N
        2.2.3. Swap (State[i], State[j])
        2.2.4. Generated Key = State[(State[i] + State[j]) mod N]
        2.2.5. A[k] [Generated Key] [i1] = A[k] [Generated Key] [i1] + 1
        2.2.6. Deducting new key with 16 bytes from each generated key to be a new secret key.
        2.2.7. k = Generated Key
        2.2.8. i1 = (i1 + 1) mod N
3. Return A[k] [Generated Key] [i1]
```

Figure 6 shows the distribution of (Z_r, Z_{r+1}) for all the first 32 bytes of RC4 where Z_r = i and Z_{r+1} = i to discover possible double-byte biases. Y-axis determines the frequents of each pair of values while the X-axis contains each pair of values.

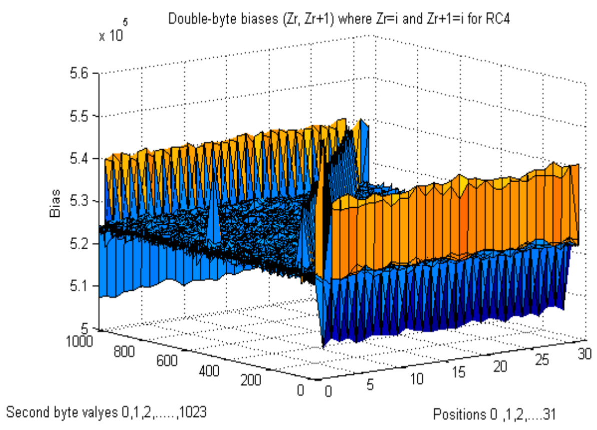

Figure 6. Double-byte biases (Z_r, Z_{r+1}) for RC4 where Z_r=i and Z_{r+1}=i.

Figure 7 shows the distribution of (Z_r, Z_{r+1}) for all the first 32 bytes of the developed RC4 with AES S-box where Z_r = i and Z_{r+1} = i.

Y-axis determines the frequents of each pair of values while the X-axis contains each pair of values.

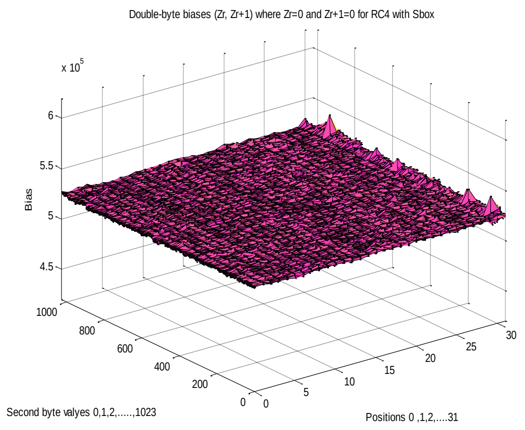

Figure 7. Double-byte biases (Z_r, Z_{r+1}) for RC4 with AES S-box where Z_r=i and Z_{r+1}=i.

8. RANDOMNESS TEST

The key stream generated by the RC4 and the developed RC4 was tested by the NIST (National Institute of Standards and Technology) Test Suite. The NIST is a statistical group for random number generator tests that consist of sixteen statistical tests to measure the randomness of output sequences of pseudo-random number generators or true random number generators, as shown below. The tests of this PRNG were done by using NIST STS-1.6.

The good random number generator likelihood was represented by the P-value in the test; this P-value was compared to 0.01. If the value is higher than 0.01, then the series is accepted, otherwise it is rejected because it shows no randomness. Some tests accepted large series sizes and failed in small series sizes, while other tests accepted both. In this paper, a large size (12 kilobyte) is generated from each secret key that has been used. These series are tested and the average of p-values results are calculated from these tests. As table I shows, the p-values are succeeded and the obtained series are uniformly distributed and random. If the tests give p-value equal to 1, then the sequence has complete randomness for this test. A p-value of zero means that the sequence has fully nonrandom.

TABLE I. RESULTS OF RUNNING NIST ON THE GENERATED KEY BY RC4 AND RC4 WITH AES S-BOX.

Test No.	**Statistical Test Name**	**RC4**		**RC4 with AES S-box**	
		P-VALUE	***Conclusion***	***P-VALUE***	***Conclusion***
1	Approximate Entropy	0.805578	SUCCESS	0.687713	SUCCESS
2	Block Frequency	0.742455	SUCCESS	0.621580	SUCCESS
3	Cumulative Sum(Forward)	0.739164	SUCCESS	0.464227	SUCCESS
4	Cumulative Sum (Reverse)	0.854066	SUCCESS	0.311231	SUCCESS
5	FFT	0.279715	SUCCESS	0.913344	SUCCESS
6	Frequency	0.898580	SUCCESS	0.481208	SUCCESS
7	Lempel-Ziv compression	0.889521	SUCCESS	0.453945	SUCCESS
8	Linear Complexity	0.407918	SUCCESS	0.842261	SUCCESS
9	Longest Runs	0.767817	SUCCESS	0.913467	SUCCESS
10	Non periodic Templates	0.540708	SUCCESS	0.570862	SUCCESS
11	Overlapping Template	0.497550	SUCCESS	0.597580	SUCCESS
12	Random Excursions	0.528198	SUCCESS	0.402825	SUCCESS
13	Random Excursion Variant	0.525591	SUCCESS	0.497233	SUCCESS
14	Rank	0.610871	SUCCESS	0.321188	SUCCESS
15	Runs	0.115965	SUCCESS	0.903451	SUCCESS
16	Serial	0.646168	SUCCESS	0.763967	SUCCESS
17	Universal Statistical	0.380374	SUCCESS	0.074774	SUCCESS

"Success" indicates that the series is acceptable and has good randomness, while "Failure" indicates that the series is not acceptable and not random.

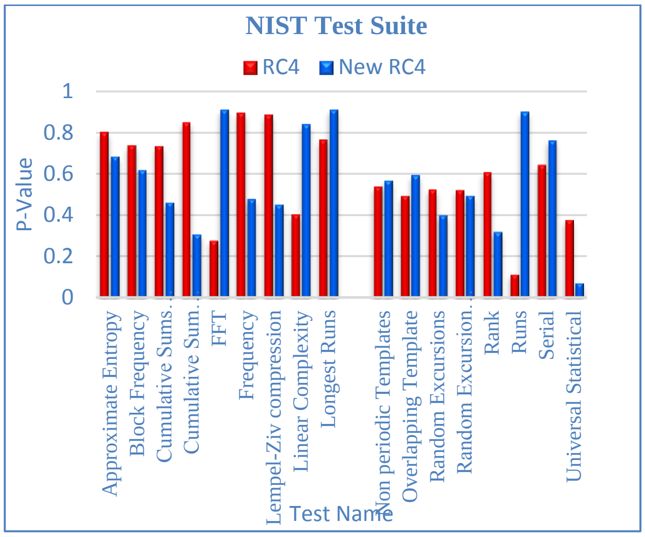

Figure 8. Results of running the NIST suite on the generated key by the RC4 and the proposed RC4.

When the implementation of the proposed algorithm takes place on the same size of the secret keys, the developed algorithm is faster than the implementation of the AES and it requires less time than that required for the RC4, as Table II shows.

TABLE II. KEY GENERATION TIME FOR THE RC4 AND THE PROPOSED RC4

Key Size	RC4 Time (m.s.)	RC4 with AES S-Box Time
301 bytes	1004	1008
3 KB	1033	1051
300 KB	35557	37223
3 MB	3202309	3421215

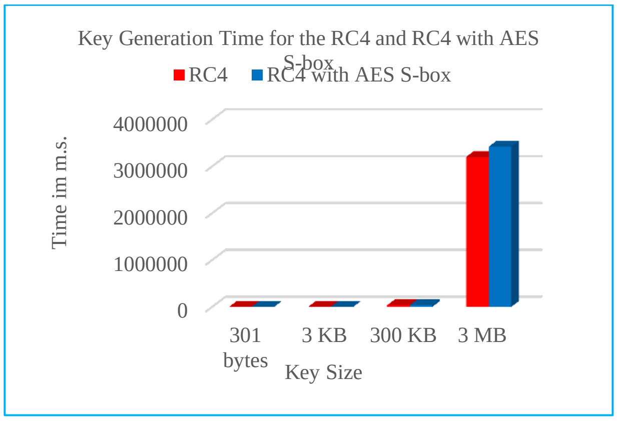

Figure 9. Key generation time for RC4 and RC4 with AES S-box.

9. CONCLUSIONS

In this paper, a new algorithm is proposed as a development for RC4 algorithm. RC4 is one of the most important symmetric cryptographic algorithms. The key generation phases are weak in key stream distribution bytes that biased toward different values. The proposed algorithm used the S-box of the AES algorithm to combine the speed of the RC4 algorithm and the robustness of the AES algorithm. The analysis of the RC4 and of the developed RC4 algorithm highlighted that the new algorithm has no single and double bias in the key stream. The developed algorithm requires little time more than that required for the RC4 by using additional swapping operations, but it is faster than the AES. The developed algorithm has passed all the statistical tests in the NIST suite, and can be used in different protocols such as SSL, WEP, and WPA protocol. As a future work, parallel processors may be used for implementing the analysis of RC4 and the developed algorithm with more key stream bytes (256 and more), and may compare the proposed scheme with other algorithms (such as DES, 3DES).

REFERENCES

[1] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, "RC4-2S: RC4 Stream Cipher with Two State Tables," Information Technology Convergence, Lecture Notes in Electrical Engineering, Springer Science Business Media Dordrecht1, pp. 13-20, DOI: 10.1007/978-94-007-6996-0_2, 2013.

[2] N. J. Al-Fardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. Schuldt, "On the Security of RC4 in TLS and WPA," In Presented as Part of the 22nd USENIX Security Symposium,13, pp. 305-320, 2013.

[3] M. McKague, "Design and Analysis of RC4-Like Stream Ciphers," (Master Thesis), University of Waterloo, Canada, Ontario, 2005.

[4] S. Sarkar, S. S. Gupta, G. Paul, & S. Maitra, "Proving TLS-attack related open biases of RC4," Designs, Codes and Cryptography, vol. 77, no. 1, pp. 231-253, 2015.

[5] S. Maitra, & G. Paul, "New form of permutation bias and secret key leakage in keystream bytes of RC4," In Fast Software Encryption, vol. 5086, pp. 253-269, 2008. Springer Berlin Heidelberg.

[6] L. L. Khine, A New Variant of RC4 Stream Cipher. Mandalay Technological University Mandalay 05052, Mandalay, Myanmar: World Academy of Science, Engineering and Technology, 2009.

[7] P. Prasithsangaree, and P. Krishnamurthy, "Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs," In Proceedings of Global Telecommunications Conference, IEEE, vol. 1443, no. 3, pp. 1445-1449, December 2003.

[8] S. Maitra, and G. Paul, "Analysis of RC4 and Proposal of Additional Layers for Better Security Margin," Lecture Notes in Computer Science, International Conference on Cryptology, vol. 5365, pp. 27-39, 2008.

[9] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, "RC4 stream cipher with a random initial state," Proceedings in 10th FTRA International Conference on Secure and Trust Computing, data management, and Applications, Lecture Notes in Electrical Engineering, pp. 407-415, 2013, Springer Netherlands.

[10] S. Paul, and B. Preneel, "Analysis of Non Fortuitous Predictive States of the RC4 Keystream Generator," Springer Computer Science, vol. 2904, pp. 52-67, 2003.

[11] M. A. Orumiehchiha, J. Pieprzyk, E. Shakour, and R. Steinfeld, "Cryptanalysis of RC4 (n, m) Stream Cipher," In Proceedings of the 6th International Conference on Security of Information and Networks, vol. 178, pp. 165-172, 2013.

[12] P. Sepehrdad, S. Vaudenay, & M. Vuagnoux, "Discovery and Exploitation of New Biases in RC4," In Selected Areas in Cryptography, vol. 6544, pp. 74-91, 2011. Springer Berlin /Heidelberg.

[13] M. M. Hammood, and K. Yoshigoe, "Previously overlooked bias signatures for RC4," In Proceedings of International Symposium on Digital Forensic and Security, pp. 101-106, 2016. doi: 10.1109 / ISDFS.2016.7473526. IEEE.

[14] M. Robshaw, and O. Billet, "New Stream Cipher Designs: The eSTREAM Finalists," Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 4986, pp. 244–266, 2008, Springer, Heidelberg.

[15] S. Mister, and S. Tavares, "Cryptanalysis of RC4-like Ciphers," In Selected Areas in Cryptography, vol. 1556, pp. 131-143, 1999, Springer Berlin/Heidelberg.

[16] A. Roos, "A class of weak keys in the RC4 stream cipher," South African, Vironix Software Laboratories: Westville, 1995. Available at http:// http://marcel.wanda.ch/Archive/WeakKeys.

[17] I. Mantin and A. Shamir, "Practical Attack on Broadcast RC4," In Fast Software Encryption, Lecture Notes in Computer Science, Springer, vol. 2355, pp. 152-164, 2002.

[18] S. R. Fluhrer, and D. A. McGrew, "Statistical Analysis of the Alleged RC4 Keystream Generator," Lecture Notes in Computer Science, Springer- Berlin Heidelberg, vol. 1978, pp. 19-30, 2001.

[19] S. M. Searan, A. M. Sagheer, and M. M. Hammood, "Analyzing of RC4 Algorithm Based on Its Single and Double Byte Bias by Using New Algorithms," International Conference on Change, Innovation, Informatics and Disruptive Technology, London – UK, 11-12 OCT, 2016, Available at http://sriweb.org/londonconf/.

Ali M. Sagheer is a Professor in the Computer College at Al-Anbar University. He received his B.Sc. in Information System (2001), M.Sc. in Data Security (2004), and his Ph.D. in Computer Science (2007) from the University of Technology, Baghdad, Iraq. He is interested in the following fields; Cryptology, Information Security, Number Theory, Multimedia Compression, Image Processing, Coding Systems, and Artificial Intelligence. He has published many papers in different scientific journals.

Sura M. Searan has received her B.Sc. in Computer Science (2013) and M.Sc. in Information Security (2016) from the College of Computer Sciences and Information Technology at University of Anbar, Baghdad, Iraq. She is interested in the following fields: Cryptology, Information Security, and Coding Systems.

Salih S. Salih has received his B.Sc. in Computer Science (2012) and M.Sc. in Computer Science (2016) from the College of Computer Sciences and Information Technology at University of Anbar, Baghdad, Iraq. He is interested in the following fields: Coding Systems, Database, and Data Mining.