I. J. Computer Network and Information Security, **2016, 12, 1-8**
Published Online December 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2016.12.01

Modification of RC4 Algorithm by using Two State Tables and Initial State Factorial

Sura M. Searan
University of Anbar/ Department of Computer Science, Baghdad, 10012, Iraq.
E-mail: Surasms917@gmail.com

Ali M. Sagheer
University of Anbar/ Department of Computer Science, Baghdad, 10012, Iraq.
E-mail: ali.m.sagheer@gmail.com

Abstract—RC4 algorithm is one of the most significant stream and symmetric cryptographic algorithms, it is simple and used in various commercial products, it has many weaknesses such as a bias in the key stream that some key bytes are biased toward some values. In this paper, a new algorithm is proposed by using initial state factorial to solve the correlation issue between public known outputs of the internal state by using an additional state table with the same length as that of the state to contain the factorial of initial state elements. The analysis of RC4 and developed RC4 algorithm is done based on their single bias and double byte bias and shows that many keystream output bytes of RC4 are produced key stream bytes that are biased to many linear combinations while developed RC4 key bytes have no single and double biases. The results show that the series that is generated by developed RC4 is more random than that generated by RC4 and the developed algorithm is faster than RC4 execution time and requires less time. Additionally, the developed algorithm is robust against many attacks such as distinguishing attack.

Index Terms—RC4, KSA (Key Scheduling Algorithm), PRGA (Pseudo-Random Generation Algorithm), Single Bias, Double Bias.

I. INTRODUCTION

Encryption is a process that involves transforming plaintext into ciphertext in order to hide its meaning and to prevent unauthorized parties from retrieving plaintext [1]. The cryptographic algorithms are designed to provide lower size, high speed of implementation, less complexity, and a larger degree of security for resource-constrained devices [2]. The strength of stream ciphers is the random key stream that guarantees secure computation of the cipher [3]. The cryptanalysis of stream cipher essentially focuses on identifying non-random process [4]. When the key size is small, it must be very efficient and encryption time be very fast, many encryptions that are used in wireless devices are based on symmetric key encryption such as RC4 algorithm [5]. RC4 is an effective stream cipher algorithm that is most popular. It is used in Oracle, SQL, Secure Sockets Layer, and Wired Equivalent Privacy Protocol [6]. The attack on this algorithm was presented by Fluhrer, Mantin, and Shamir, it is an algorithm to use the symmetric key and it is an important one of the encryption algorithms [7]. This algorithm includes two main components to generate the key, the first is (KSA) Key Scheduling Algorithm and the other is (PRGA) Pseudo-Random Generation Algorithm [8]. RC4 starts with the permutation and uses the secret key with a variable length from 1 to 256 bits against a 256-bit state table [9]. The key is limited to 40 bits because of missing of restrictions, but it is sometimes used as 128 key bits [10]. Symmetric encryption can be classified into stream and block ciphers [11]. RC4 is analyzed by different people and there are different weaknesses detected [12]. KSA is more problematic and it is prepared to be simple [13]. At the beginning, few bytes of the output of PRNG are biased or related to some key bytes [14]. There are different types of attack that are classified by the amounts of information available to the adversary for cryptanalysis based on available resources [15]. The aim of this work is to solve interconnection between public known outputs of the internal state of RC4.

The rest of the paper is arranged as follows. Section 2 gives the related works. Section 3 shows a brief depiction of RC4 algorithm. Several weaknesses of RC4 are determined in section 4. Section 5 illustrates the Modified RC4 by using Two State Tables and Initial State Factorial. Sections 6 and 7 show the implementation, results and discussion. In the results, section A shows a comparison between analyzing of RC4 and developed RC4 with factorial based on single byte bias. And section B shows a comparison between analyzing of RC4 and developed RC4 with factorial based on double byte bias. Conclusion is shown in section 8.

II. RELATED WORKS

Mantin I. and Shamir A. (2001) showed an essential statistical weakness in the RC4 keystream by analyzing RC4 algorithm. This weakness makes it insignificant to discriminate between random strings and short outputs of RC4 by analyzing the second bytes. It is observed that

the second output byte of RC4 has a very strong bias that takes the value 0 with twice the expected likelihood (1/128 instead of 1/256 for n = 8). The main result is the detection of a slight distinguisher between the RC4 and random ciphers, that needs only two output words under many hundred unrelated and unknown keys to make robust decision [1]. Sepehrdad P. *et al.* (2011) discovered new biases in RC4 and proposed a mechanism to focus on linear attachments in the RC4 and detected linear attachment in the PRGA of the RC4 algorithm, and presented the elements within one trial of PRGA. Then this way is popularized to RC4 as a black crate with confident keywords as an input and words of the keystream as an output. These mechanisms lead to the detection of 57 attachments in the RC4. Some of these can be immediately utilized in the present key retrieval attacks on the RC4, WEP, and WPA [3]. Al-Fardan N. J. *et al.* (2013) measured the security of RC4 in TLS and WPA and analyzed RC4 based on its single and double byte bias and attacked it based on its two types of bias by using plaintext recovery attack. Their results show that there are biases in the first 256 bytes of the RC4 keystream that can be exploited by passive attacks to retrieve the plaintext by using 2^{44} random keys. They focused on the multi-session setting, where the same plaintext is repeatedly encrypted with different keys and could recover full plain text from cipher text by using single byte bias attack and double byte bias attack [12]. Hammood M. M. *et al.* (2015) presented enhancing RC4 security and speed. Many algorithms were proposed as development for RC4. The first is RRC4 (RC4-Random initial state) which is to make RC4 more secure by increasing its randomness. The second suggestion is RC4 with two state tables to increase the randomness in the key sequence and the execution time of RC4-2State is faster than that of RC4. The last suggestion is RC4-2State + which is to produce 4 keys in every cycle to improve the randomness in the key sequence. The output sequences of all suggested algorithms provide more randomness [13].

III. RC4 Description

The RC4 algorithm was proposed by Ron Rivest in1987 and kept secret as a trade secret until it was leaked in 1994. It is very fast and simple in design [16]. The internal state is an array S of (2n words) [17]. RC4 has a variable length of the key that ranges between (0 - 255) bytes for initializing an array of 256 bytes in the initial state (State [0] to State [255]) [2].RC4 is carried out in two phases: The first is the key scheduling algorithm (KSA). It initializes the internal state [7].

RC4 starts the replacements and uses private key to get a random replacement with the KSA .Based on confidential key, the other phase is PRGA that produces key bytes that are XOR-ed with original bytes to get the cipher [5] .The state array is used to produce pseudo-random bits .These are done in the KSA .The operation which is performed between key and plain text is equivalent in some regard to the Vernam cipher [6] .The key is limited to 40 bits but it is sometimes used as a 128 bit key [8] .It has the receptivity to use 1 to 2048 key bits .In general, the RC4 key size is "5 to 16 bytes" (40 to 128 bits) and the size of the typical state is 256 bytes [7]. It is highly utilized on the internet. It is used popularly as a default cipher for "Secure Socket/Transport Layer Security" (SSL/TLS) connections. It is very fast and simple in design and it is a set of stream systems represented by n that shows the word size in bits. The inner state is array (State) of (2n words) [8].

Algorithm 1. KSA

```
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 step is pseudo-random generation algorithm. It generates the output keystream

Algorithm 2. PRGA

```
INPUT: State, Plaintext i
OUTPUT: Key sequence (K sequence)
1. Initialization:
        1.1     i = 0
        1.2     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 XOR-ed with the Plaintext

$$C_i = K_i \oplus \text{Plaintext i [4].}$$

IV. The Weakness of RC4

There are several weaknesses found in RC4 algorithm. Some of these weaknesses are easy and can be resolved, but other weakness is dangerous because attackers can exploit it. Another weakness in initialization state is a statistical bias which occurs in distributing words of the first output [7]. The key stream which begins the algorithm swaps the entry of the s-box exactly one time (identical to the pointer i that points to an entry) for low values of i, it is probable that $S_j = {}_j$ during the initialization [18]. Roos found weaknesses in RC4 that has serious correlation between generated value and the first few values of the state table. The first byte of the generated key is highly correlated with a few key bytes.

So, the keys allow precursor of first bytes from the output of the PRGA [19]. The goal of the attack is to retrieve the original key, the internal state, or the output keystream to have an access to the original messages.

From the previous studies based on KSA and PRGA, there are some weaknesses of RC4 such as the biased bytes, distinguishers, key collisions, and key recovery from the state [20]. PRGA is reversible in nature. Then it is very simple to retrieve the secret key from the state [21]. Mantin and Shamir detected the major weakness in the second round the probability of zero output bytes. Fluher found a large weakness, if anyone knows the portion of the private key then its potential is to attack the RC4. Paul and Maitra found the secret key by using initial state table [22]. There are various methods of applying a brute force attack to the RC4 that is composed of two types: KSA attacks and PRGA attacks [23]. The vulnerabilities include A broadcast attack by using various unique keys for the encryption on similar plaintext offers the redundancy of the cipher text. The vulnerabilities include that PRGA is utilized as a seed for the RC4 which can be weak by analyzing a large collection of cipher text and as a result got duplicated encryption of the same plaintext with the same key [24]. For making RC4 secure and capable of standing against the attack, lot of researches was done over RC4 to enhance the security of RC4.

V. The Modified RC4 by Using Two State Tables and Initial State Factorial

RC4 has various weaknesses in the KSA and PRGA that cause vulnerablity to this algorithm. This section determines the new insertion to the RC4 algorithm to improve it and to solve the weak key problem by using two state tables, one of them contains the factorial of other state contents with the same length to reduce the weakness that is exploited by the attacks. This algorithm consists of initialization step (KSA) and another step (PRGA) as shown in Algorithms (3) and (4). All addition operations are implemented in mod state length (N). The first step (KSA) takes a secret key k with a variable length between 1 and 256 n-bit words. In the first step of the KSA, one of the state tables is filled by the factorial of the contents of the other state table that is generated by the sender and filled with numbers from 0 to N-1. The input is the secret key used as a state table seed. After the KSA, the state becomes input to the next step (PRGA). In the PRGA step, additional operations are used as permutation to the state table. This phase generates the keystream that is XOR-ed with the plaintext to get the ciphertext.

Algorithm 3. KSA of developed RC4 with Factorial

```
INPUT: Key[i].
OUTPUT: State [i].
1.      For (i = 0 to 255)
                State[i] = i.
2.      State_Fact[i] = Factorial(i) mod 256
3.      j = 0
4.      For (i = 0 to 255)
        4.1     j = (State_Fact[i] + State[i] + Key [i mod
key-length]) mod 256
        4.2     Swap (State[i], State[j])
5.      Output: State [i].
```

The second is PRGA which generates the output keystream:

Algorithm 4. PRGA of developed RC4 with Factorial

```
INPUT: State [i], Plain i.
OUTPUT: Key sequence (Key seq.)
1.      Set     i = 0, j = 0
2.      Output Generation loop
        2.1     i = (i + 1) mod 256
        2.2   j = (State [(j + state[i] ) mod 256 ]) mod 256
        2.3     Swap (State[i], (State-Fact [j] mod 256))
        2.4     Z = (State[(i + j) mod 256] + State[(j +
State[state[i]]) mod 256]) mod 256
        2.5     Key sequence = State [Z]
3.      Output: Key sequence.
```

$$\text{Cipher } i = \text{Key seq.} \oplus \text{Plain } i.$$

VI. Implementation

This algorithm is executed by using C# language. The inputs to this algorithm are an initial state that is filled with the values from 0 to 255 and secret key with a length between 1 and 256, and another state table with 256 bytes to contain the factorial of initial state elements. The implementation of the proposed algorithm required less time than that required for implementation of RC4 when implemented on the same size of secret keys and showed that the proposed algorithm is faster than RC4 as shown below. The table 1 and figure 1 below show the time of key generation for RC4 and the proposed algorithm

Table 1. Key Generation Time for RC4 and Developed RC4 with Initial State Factorial.

Key size	RC4 Time (m. s.)	RC4 with Factorial Time (m.s.)
1 kilobytes	4185	4091
2 kilobytes	4237	4191
3 kilobytes	4711	4213
5 kilobytes	6899	6372

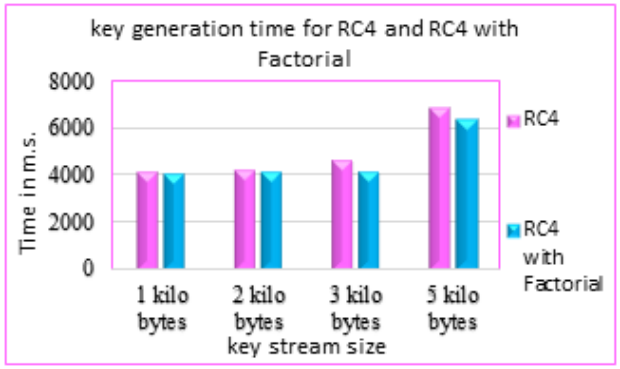

Fig.1. Implementation Time of RC4 and Developed RC4.

VII. Results and Discussion

The generated key stream is examined by NIST (National Institute of Standards and Technology) Test Suite that is a statistical combination for random number generator test that includes 16 statistical tests for measuring the output series randomness of pseudo-random number or true random number generators .The tests of this PRNG were done by using NIST STS-1.6 .The likelihood of a good random number generator is represented by P-value .Some tests accepted large sequence sizes and failed in the small sequence size, and other test accepted both large and small sizes .In our program, a large size (2,000,000 bits) is generated from each secret key .these sequences are tested, and p-values average which resulted from these tests are calculated as shown in table 2, in the test, P-value is compared to 0.01, the p-values are passed when they are greater than 0.01, and the produced series is random, and uniformly distributed .If the tests give p-value equal to 1, then the series is taken to have complete randomness. a p-value of zero means that the sequence is fully nonrandom. The SUCCESS means that the sequence is acceptable and it has good randomness, where FAILURE indicates that it is not acceptable and not-random. In general, these sixteen tests are composed of two collections. The first is called non-parameterized test and include Frequency Test, Cumulative Sums Test (forward and reverse), Discrete Fourier Transform (Spectral) Test, Lempel-Ziv compression Test, test for Longest Run of Ones in a Block, Rank Test, Runs Test, Random Excursions Test, and Random Excursions Variant Test. The second collection is called a parameterized test, it includes Serial Test, Linear Complexity Test, Overlapping Template of All One's Test, Non-overlapping Template Matching Test, Approximate Entropy Test, Block Frequency Test, and Universal Statistical Test.

Table 2. Result of Running NIST on the Generated Key by RC4 and the Proposed RC4.

Test No.	Statistical Test Name	RC4		RC4 with Factorial	
		P-VALUE	Conclusion	P-VALUE	Conclusion
1	Approximate Entropy	0.805578	SUCCESS	0.195979	SUCCESS
2	Block Frequency	0.742455	SUCCESS	0.990906	SUCCESS
3	Cumulative Sums (Forward)	0.739164	SUCCESS	0.829138	SUCCESS
4	Cumulative Sum (Reverse)	0.854066	SUCCESS	0.716066	SUCCESS
5	FFT	0.279715	SUCCESS	0.556777	SUCCESS
6	Frequency	0.898580	SUCCESS	0.582269	SUCCESS
7	Lempel-Ziv compression	0.889521	SUCCESS	0.730735	SUCCESS
8	Linear Complexity	0.407918	SUCCESS	0.828157	SUCCESS
9	Longest Runs	0.767817	SUCCESS	0.985925	SUCCESS
10	Non periodic Templates	0.5407084	SUCCESS	0.527034	SUCCESS
11	Overlapping Template	0.497550	SUCCESS	0.614743	SUCCESS
12	Random Excursions	0.528198	SUCCESS	0.593593	SUCCESS
13	Random Excursion Variant	0.525591	SUCCESS	0.472056	SUCCESS
14	Rank	0.610871	SUCCESS	0.144541	SUCCESS
15	Runs	0.115965	SUCCESS	0.574394	SUCCESS
16	Serial	0.646168	SUCCESS	0.583851	SUCCESS
17	Universal Statistical	0.380374	SUCCESS	0.580536	SUCCESS

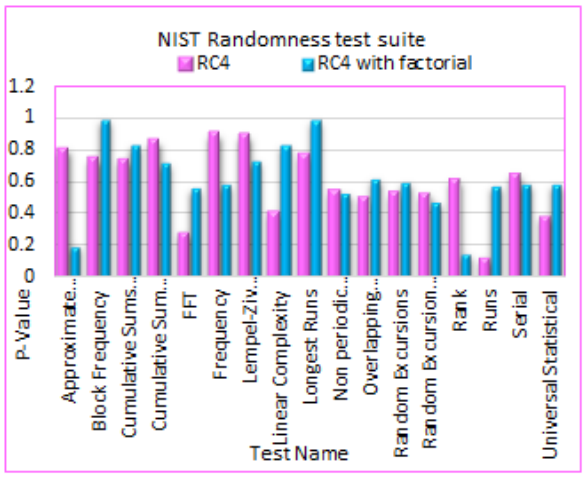

Fig.2. NIST Statistical Test for RC4 and Developed RC4.

A. Comparison Between Analyses of RC4 and Developed RC4 with Factorial Based on Single Byte Bias.

RC4 has many weaknesses in the generated key stream, the key stream bytes are biased. In 2002 Mantin and Shamir [1] found that in the second round, the key is biased toward zero with high probability. AL-Fardan *et al.* also Hamood M.M. *et al.* analyzed RC4 based on its bias and proved that the first 256 bytes are biased. In this work, RC4 and the new algorithm were also analyzed and proved that developed RC4 has no bias while RC4 key stream is biased and shows the same bias that is proved in the literature, this work reduces the search space and uses state with length 32 and 2^{30} random secret keys each one with length 16 and proved bias in standard time (less than one hour) as shown below. Expected biases start appearing for runtime 2^{21} key generation. No bias is statistically identified for the proposed algorithm. Single-bias is calculated by using the following algorithm:

Algorithm 5. Measuring Distributions of RC4 Keystream Bytes Based on Its Single Bias

INPUT: K [k_1, k_2, …., k_{16}].
OUTPUT: Key position (K_p), key value (K_v), and the number of frequencies (K_f) for each position of key stream bytes.
1. For (x = 1 to 2^{34}) Do
   1.1 i = 0, j = 0
   1.2 Call Algorithm 1: KSA.
   1.3 Call Algorithm 2: PRGA.
   1.4 Deducting new key with a length of 16 bytes from each generated key are to be new secret key.
2. For (col = 0 to key Length)
   2.1 For (row = 0 to 2^{34})
      Set key [row] [col] as string
   2.2 For (i = 1 to values. Count)
      2.2.1 If (values [i] = value)
      2.2.2 Increment count by 1
      2.2.3 Key position = col
      2.2.4 Key value = value
      2.2.5 Number of frequencies = (count / (2^{34} * 16))
3. Output: Kp, Kv, and Kf for each position of key stream bytes.

This algorithm is implemented in C# language. Several biases were identified in the previous researches .RC4 is successfully reproduced and proved these bias in the first 32 bytes of keystream while developed RC4 has no bias in the first 32 positions of keystream byte. The algorithm of measuring key distribution bytes is implemented with 2^{34} secret key as shown in figures 3,4,5 and 6.

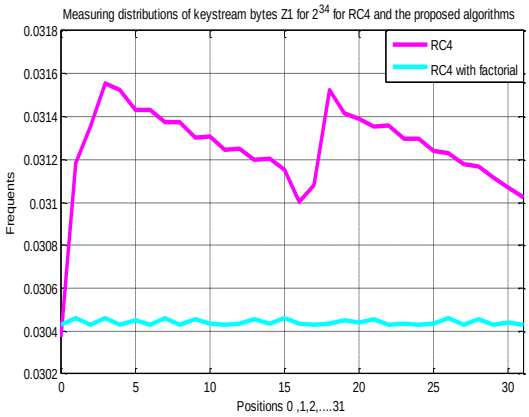

Fig.3. Key Distribution in the 1st Position with 2^{21} for RC4 and Developed RC4.

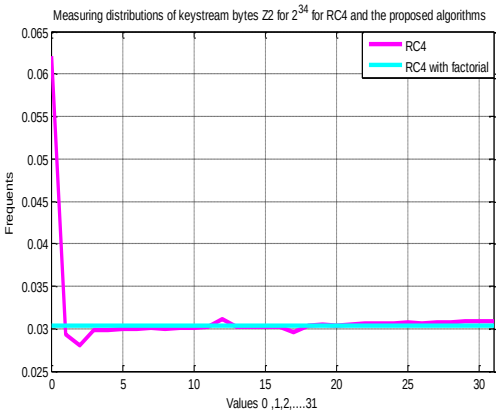

Fig.4. Key Distribution in the 2nd Position with 2^{21} for RC4 and Developed RC4.

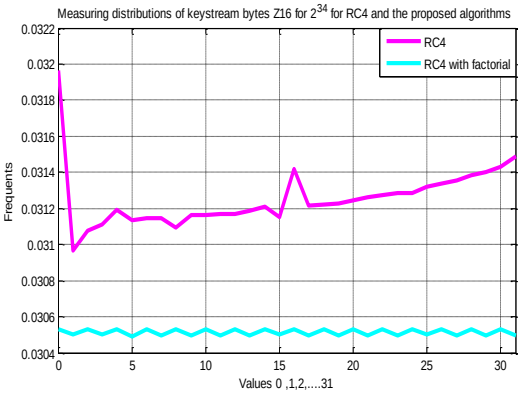

Fig.5. Key Distribution in the 16th Position with 2^{21} for RC4 and Developed RC4.

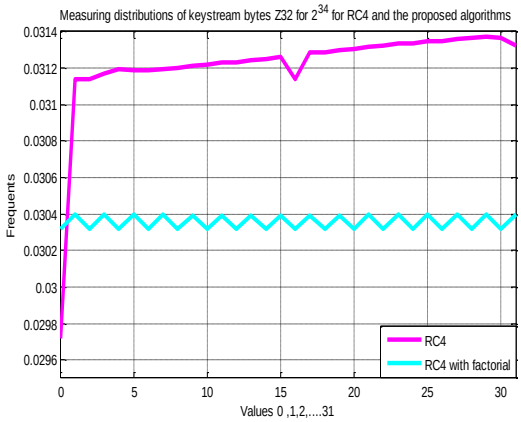

Fig.6. Key Distribution in the 32nd Position with 2^{21} for RC4 and Developed RC4.

B. Comparison Between Analyses of RC4 and Developed RC4 with Factorial Based on Double Byte Bias.

After explaining single-byte biases that are of great benefit to the cryptographic society, the attack simply can be avoided by ignoring the initial bytes. Thus, RC4 with additional configuration can still resist single-byte bias attack. However, the researchers have studied and investigated biases beyond initial bytes and different multi-byte biases have been discovered in the key stream of RC4. Fluhrer and McGrew [25] 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 RC4. Hamood et al. [26] 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 biases of Fluhrer and McGrew and their work was endorsed by AlFardan *et al.* [12] They found two new positive biases not mentioned in [25] by Fluhrer and McGrew.

This work reproduced the Fluhrer and McGrew biases and Hammood 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 proposed algorithms does not generate any statistical bias and its output in the range only $\pm 2^4$ from the predicted occurrences. Algorithm 6 below is designed to determine the measure of double byte bias. The main idea of this algorithm is to measure the appearance of the consecutive pair (Z_i, Z_{i+1}) in each position of the output of RC4. The measure of double byte bias is illustrated in the algorithm below:

Algorithm 6. Measuring Distributions of RC4 Keystream Bytes Based on Its Double Bias

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 1. KSA
   2.2 For (x = 1 to 2^{32})
      2.2.1 i = (i + 1) mod 256
      2.2.2 j = (j + State[i]) mod 256
      2.2.3 Swap (State[i], State[j])
      2.2.4 Generated Key = State[(State[i] + State[j]) mod 256]
      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 new secret key.
      2.2.7 k = Generated Key
      2.2.8 i1 = (i1 + 1) mod 256
3. Output: A[k][Generated Key][i1].

The figure below shows the distribution of (Zr, Zr+1) for all the first 32 bytes where Zr = i and Zr+1 = i for RC4.

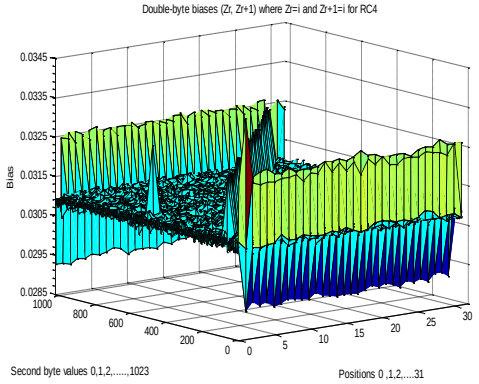

Fig.7. Double-Byte Biases (Zr, Zr+1) for RC4 where Zr=i and Zr+1=i.

The figure below shows the double-byte biases (Z_r, Z_{r+1}) for modified RC4 with factorial where Z_r=i and Z_{r+1}=i for the first 32 bytes.

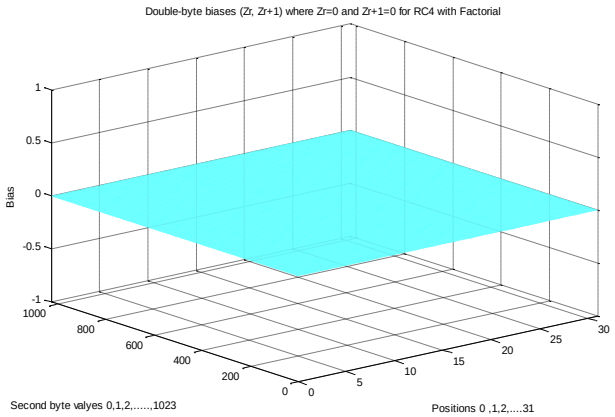

Fig.8. Double-Byte Biases (Z_r, Z_{r+1}) for Modified RC4 with Factorial where Z_r=i and Z_{r+1}=i.

VIII. Conclusion

RC4 stream cipher is a significant encryption algorithm and it is one of the widely used cryptosystems on the Internet that is used to keep information privacy. RC4 implementation is simple and fast compared with other encryption algorithms, but its key bytes are biased, this weakness makes RC4 vulnerable to attack. The proposed algorithm uses factorial of the state table contents and addition operations in KSA and PRGA to increase the randomness of the generated key while the key generation time of suggested algorithm is faster than key generation time of RC4. The key stream of the proposed algorithm has no single or double bias in the first 32 position as we reduce the search space and measured distribution bytes in standard time while RC4 key stream is biased in different positions. The generated key stream of the proposed RC4 has passed the NIST suite of statistical tests. Thus, it can be executed in the software or hardware.

References

[1] I. Mantin and A. Shamir, "A Practical Attack on Broadcast RC4". Springer, Lecture Notes in Computer Science. 2002, (2355), pp 152-164.

[2] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, "RC4-2S: RC4 Stream Cipher with Two State Tables". Springer, Lecture Notes in Electrical Engineering, 2013, 1, pp 13-20.

[3] P. Sepehrdad, S. Vaudenay, and M. Vuagnoux, M. "Discovery and Exploitation of New Biases in RC4", Springer, In Selected Areas in Cryptography, 2011, pp. 74-91.

[4] M. E. McKague, "Design and analysis of RC4-like stream ciphers", MS.C. Thesis, University of Waterloo, Canada/ Ontario, 2005.

[5] P. Prasithsangaree, and P. Krishnamurthy, "Analysis of energy consumption of RC4 and AES algorithms in wireless LANs", In Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE, 2003, 3, pp 1445-1449.

[6] A. M. S. Rahma, A. M. Sagheer, and A. A. Salih, "Development of RC4 Stream Ciphers Using Boolean Functions", Journal of Baghdad College of Economic Sciences University, 2012, 29.

[7] L. Stosic, and M. Bogdanovic, "RC4 stream cipher and possible attacks on WEP", Editorial Preface, 2012, 3(3).

[8] S. Maitra, and G. Paul, "New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4", In Fast Software Encryption, Springer, 2008, pp. 253-269.

[9] S. Maitra, and G. Paul, "Analysis of RC4 and Proposal of Additional Layers for Better Security Margin", Springer, Lecture Notes in Computer Science. 2008, pp 27-39.

[10] C. Garman, K. G. Paterson, and T. Van der Merwe, "Attacks Only Get Better: Password Recovery Attacks Against RC4 in TLS", In Presented as Part of the 24th USENIX Security Symposium (USENIX Security 15), 2015.

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

[12] 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. USENIX, 2013, 13, pp 305-320.

[13] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, "Enhancing Security and Speed of RC4", International Journal of Computing and Network Technology, 2015, 3(2).

[14] L. L. Khine, "A New Variant of RC4 Stream Cipher", Mandalay, Myanmar: World Academy of Science, Engineering and Technology, 2009.

[15] M. U. Bokhari, S. Alam, F. S. Masoodi, "Cryptanalysis Techniques for Stream Cipher: a survey", International Journal of Computer Applications, 2012, 60(9), pp 29-33.

[16] K. K. H. Wong, G. Carter, and E. Dawson, "An analysis of the RC4 Family of Stream Ciphers Against Algebraic Attacks", In Proceedings of the Eighth Australasian Conference on Information Security, 2010, 105, pp 67-74.

[17] 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, 2013, pp 165-172.

[18] S. Mister, and S. Tavares, "Cryptanalysis of RC4-like Ciphers", In Selected Areas in Cryptography, 1999, pp 632-632. Springer.

[19] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, "RC4 Stream Cipher with a Random Initial State", Springer, Lecture Notes in Electrical Engineering, 2013, 1, pp 407-416.

[20] P. Jindal, and B. Singh, "A Survey on RC4 Stream Cipher", International Journal Computer Network and Information Security, 2015, 7, pp 37-45.

[21] M. Robshaw, and O. Billet, "New Stream Cipher Designs: The eSTREAM Finalists", Springer, Lecture Notes in Computer Science, 2008.

[22] P. Pardeep, and P. K. Pateriya, "PC 1-RC4 and PC 2-RC4 Algorithms: Pragmatic Enrichment Algorithms to Enhance RC4 Stream Cipher Algorithm", International Journal of Computer Science and Network, 2012, 1(3).

[23] M. Omari, and H. S. Soliman, "Exponential Brute-Force Complexity of a Permutation Based Stream Cipher", International Journal Computer Network and Information Security, 2013, 1, pp 1-13.

[24] V. K. Keerthi, and R. P. Arun, "Taxonomy of SSL/TLS Attacks", International Journal Computer Network and Information Security, 2016, 2, pp 15-24.

[25] S. R. Fluhrer, and D. A. McGrew, "Statistical Analysis of the Alleged RC4 Keystream Generator", Springer, Lecture notes in computer science, 2001, (1978), pp 19-30.

[26] M. M. Hammood, and K. Yoshigoe, "Previously Overlooked Bias Signatures for RC4", International Symposium on Digital, Forensic Security, 2016, 101-106. doi:10.1109.

Authors' Profiles

Sura M. Searan has received her B.Sc. in Computer Science (2013) from the University of Anbar, Iraq. She is a master student (2014, till now) in the Computer Science Department, College of Computer Sciences and Information Technology at Al-Anbar University. She is interested in the following fields; Cryptology, Information Security, Coding Systems.

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.

How to cite this paper: Sura M. Searan, Ali M. Sagheer,"Modification of RC4 Algorithm by using Two State Tables and Initial State Factorial", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.12, pp.1-8, 2016.DOI: 10.5815/ijcnis.2016.12.01

