JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31st December 2016, Volume 1, Issue 2, JSEIS, CAOMEI Copyright © 2016
www.jseis.org

Modification of RC4 algorithm to increase its security by using mathematical operations

[1] Ali M. Sagheer, [2] Sura M. Searan, [3] Rawan A. Alsharida
[1, 2]College of Computer Sciences and Information Technology, University of Anbar, Anbar, Iraq.
[3]College of Computer Sciences and Mathematics, University of Tikrit, Tikrit, Iraq.
Email: [1] ali.m.sagheer@gmail.com, [2] Surasms917@gmail.com, [3] rawanpc@yahoo.com

ABSTRACT

RC4 algorithm is one of the most widely used stream ciphers. It is fast, simple and suitable for software and hardware. It is used in many applications, but it has a weakness in the distribution of key stream bytes, the first few key stream bytes of PRNG are biased or related to some secret key bytes and thus the analysis of key stream bytes makes it possible to attack RC4, and there is a correlation between the key stream bytes that make it weak and breakable by single and double byte bias attack. This work shows a new algorithm proposed by using initial state factorial to solve the correlation issue between public known outputs of the internal state and making this algorithm is robust against attack by using an additional state table with the same length of the state to contain the factorial of initial state elements. Also, shows the single byte bias attack on RC4 by using the newly designed algorithm. The results showed that the proposed algorithm is robust against attack and could retrieve the first 32 bytes of the plain text by using the proposed algorithm of single byte bias attack with a probability of 100%. Additionally, the developed algorithm is robust against many attacks such as distinguishing attack.

Keywords: RC4; KSA; PRGA; Single Byte Bias; Double Byte Bias; Single Byte Bias Attack;

1. INTRODUCTION

Encryption is a process that accomplishes of 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 guaranties secure computation of the cipher [3]. The cryptanalysis of stream cipher essentially focuses on identifying non-random proceeding [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 Key Scheduling Algorithm (KSA) and the other is Pseudo-Random Generation Algorithm (PRGA) [8]. RC4 starts with the permutation and uses the secret key with a variable length from 1 to 256 bits to preface 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 different weaknesses are detected [12]. KSA is more problematic and it aprepared 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.

2. LITERATURE REVIEW

Several researchers in information security analyzed RC4 algorithm based on its weakness and suggested different solutions, but the ways of bias calculations were slow, not efficient, and used a huge number of data. This section shows the previous studies related to this work: 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. And observe 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]. 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 attacking it based on its 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 244 random keys [12]. Hammood M. M. et al. (2015) presented research to enhance RC4 security and speed. Many algorithms are proposed as development for RC4. The first is RRC4 (RC4-Random initial state), presented 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 faster than RC4. The last suggestion is RC4-2State + 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].

3. RC4 CONCEPT

Many of stream cipher algorithms are based on the use of Linear Feedback Shift Registers (LFSRs) particularly in the hardware, but the design of RC4 algorithm evades the use of LFSR [16]. This algorithm consists of two main components to generate the key, the first is Key Scheduling Algorithm (KSA) and the second is Pseudo-Random Generation Algorithm (PRGA) that is implemented sequentially [17]. KSA is more problematic, it was prepared to be simple. At the beginning, few bytes of the output of PRGA are biased or attached to some bytes of the secret key; therefore, analyzing these bytes makes them probable for attacking RC4 [7]. The internal permutation of RC4 is of N bytes, it is a key. The length of the private key is typical between 5 to 32 bytes and is recurrent to form the final key stream. KSA can produce initial Permutation of RC4 by scrambling the corresponding permutation using the key. This permutation (State) in KSA is used as an input to the second step (PRGA) that generates the final key stream [11]. RC4 starts with the permutation and uses a secret key to produce a random permutation with KSA. Based on a secret key, the next stage is PRGA that generates keystream bytes which XOR-ed with the original bytes to get the ciphertext [12]. The concept of RC4 is to make a permutation of the elements by swapping them to accomplish the higher randomness. RC4 algorithm has a variable length of key between (0-255) bytes to initialize the 256 bytes in the initial state array (State [0] to State [255]) [13]. The algorithms below show KSA and PRGA steps of the RC4 algorithm:

Algorithm 1. KSA

```
INPUT: Key
OUTPUT: S
1.    For (x = 0 to 255)
      1.1   S[x] = x
2.    Set y = 0
3.    For (x = 0 to 255)
      3.1   y = (y + S[x] + Key [x mod key-length]) mod 256
      3.2   Swap(S[x], S[j])
4.    Output:  S
```

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

Algorithm 2. PRGA

```
INPUT: S, Plaintext ^x
OUTPUT: Key sequence (K sequence)
1. Initialization:
      1.1   x = 0
      1.2   y = 0
2.    For (x = 0 to Plaintext length)
      2.1   x = (x + 1) mod N
      2.2   y = (y + S[x]) mod N
      2.3   Swap(S[x], S[y])
      2.4   K sequence = S [S[x] + S[y]] mod N
3. Output:  K sequence
```

The output sequence of key *K* is XOR-ed with the Plaintext
$C_x = K_x \oplus$Plaintext x [4]

4. THE WEAKNESS OF RC4

There are several weaknesses found in RC4 algorithm. Some of these weaknesses are easy and can be resolved, but other weaknesses are dangerous that attackers can exploit it. One of these weaknesses in initialization state is a statistical bias that occurs in distributing words of the first output [7]. The key stream beginning algorithm swaps the entry of the s-box exactly one time (identical to the pointer I that points to an entry) for low values, it is probable that SJ = j during the initialization [18]. Roos [19] also found RC4 weakness that is a high attachment between the first state table values and generated values of the key stream. The essential cause is the state table that began in series (0, 1, 2, …, 255) and at least one out of each 256 potential keys, the first generated byte of the key is highly attached with a few key bytes. Thus, the keys allow precursor of the first bytes from the output of PRGA. To reduce this problem, it was proposed to ignore the first bytes of the output of 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 found the main weakness of RC4 in the second round. The likelihood of zero output bytes. Paul and Maitra found a private key by using the elementary state table and generated equations on the initial state bases and selected some of the secret key bytes on the basis of assumption and keep private key discovery by using the equation. Thus, the safeness of RC4 is based on a private key security and the internal states. Various attacks focus on getting the private key of the internal states [22]. The attack aims to retrieve the main key, internal state, or final key stream to access to the original messages [19].

5. THE MODIFIED RC4 BY USING FACTORIAL (RC4-FACT)

RC4 has many weaknesses in the KSA and PRGA that cause vulnerable to this algorithm. This section shows a new enhancement to the RC4 algorithm to improve it and to solve the weak key problem by using two arrays, one of them includes 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 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 filled by the factorial of the contents of the other state table that generated by the sender and filled by 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 used as permutation to the state table. This phase generates the keystream that X-OR-ed with the plaintext to get the ciphertext.

Algorithm 3. KSA of Modified RC4-Fact

```
INPUT: Key[x].
OUTPUT: S [x].
1.      For (x = 0 to 255)
                S[x] = x.
2.      For (x=255 to 0)
        For (c = 1 to x)
                S_Fact[x] = (S_Fact[x] *c) mod 256
3.      y = 0
4.      For (x = 0 to 255)
        4.1     y = (S_Fact[x] + S[x] + Key[x mod key-length]) mod 256
        4.2     Swap (S[x], S[y])
5.      Output:  S [x].
```

The second is PRGA which generates the output keystream:

Algorithm 4. PRGA of Modified RC4-Fact

```
INPUT: S [x], Plain x.
OUTPUT: Key sequence (Key seq.)
1.      Set     x = 0, y = 0
2.      Output Generation loop
        2.1     x = (x + 1) mod 256
        2.2    y= (S [(y + S[x]) mod 256]) mod 256
        2.3        Swap (S[x], (S-Fact [y] mod 256))
        2.4        Z = (S[(x + y) mod 256] + S[(y + S[S[x]]) mod 256]) mod 256
        2.5        Key sequence = S [Z]
3.      Output:  Key sequence.
Cipher x = Key seq. ⊕ Plain x.
```

6. 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.

Table. 1 Key generation time for RC4 and developed RC4-Fact.

Key size	RC4 Time (m. s.)	RC4-Fact Time (m.s.)
1 kilobytes	4185	4091
2 kilobytes	4184	4110
3 kilobytes	4703	4191
5 kilobytes	6421	6295

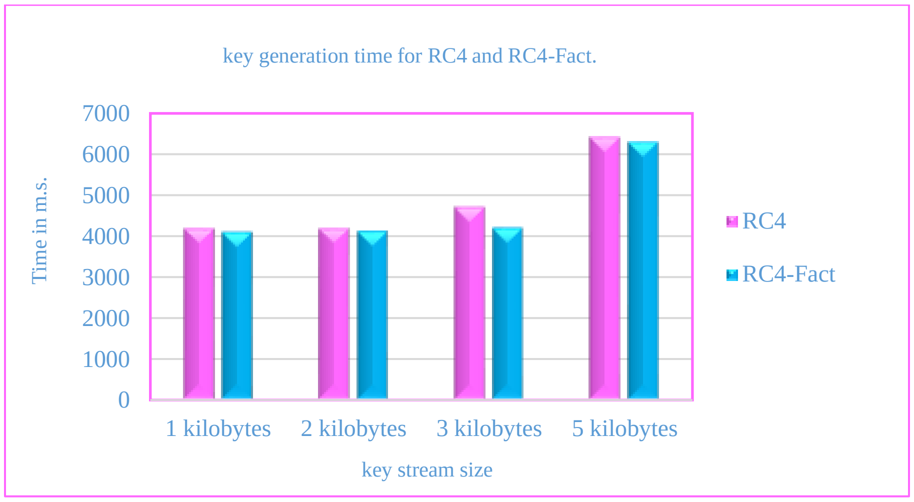

Figure. 1 Implementation Time of RC4 and Developed RC4

7. RESULTS AND DISCUSSION

The produced key stream is examined by the NIST (National Institute of Standards and Technology) test suite that is a statistical combination for the random number generator test that is composed of 16 statistical tests for measuring the output series randomness of pseudo-random number or true random number generators [23]. 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 tests accepted both large and small sizes. In our program, a large size (2,000,000 bits) is generated from each secret key. These sequences tested, and calculated p-values average result from these tests as shown in Table 2, in the test, P-value is compared to 0.01, the p-values are passed when it is greater than 0.01, and the produced series be random, and uniformly distributed. If the tests give p-value equal to 1, then the series taken to have complete randomness. A p-value of zero means that the sequence is fully non-random. The SUCCESS means that the sequence is acceptable and it has good randomness, where FAILURE indicates that it is not acceptable and not-random.

Table. 2 Result of running NIST on the generated key by RC4 and RC4-Fact.

Test No.	Statistical Test Name	RC4		RC4-Fact.	
		P-VALUE	Conclusion	P-VALUE	Conclusion
1	Approximate Entropy	0.805578	PASS	0.189928	PASS
2	Block Frequency	0.742455	PASS	0.892817	PASS
3	Cumulative Sums (Forward)	0.739164	PASS	0.838256	PASS
4	Cumulative Sum (Reverse)	0.854066	PASS	0.808187	PASS
5	FFT	0.279715	PASS	0.574878	PASS
6	Frequency	0.898580	PASS	0.911358	PASS
7	Lempel-Ziv compression	0.889521	PASS	0.821626	PASS
8	Linear Complexity	0.407918	PASS	0.799356	PASS
9	Longest Runs	0.767817	PASS	0.909724	PASS
10	Non periodic Templates	0.5407084	PASS	0.608033	PASS
11	Overlapping Template	0.497550	PASS	0.626771	PASS
12	Random Excursions	0.528198	PASS	0.593594	PASS
13	Random Excursion Variant	0.525591	PASS	0.472056	PASS
14	Rank	0.610871	PASS	0.354270	PASS
15	Runs	0.115965	PASS	0.494192	PASS
16	Serial	0.646168	PASS	0.6234565	PASS
17	Universal Statistical	0.380374	PASS	0.520501	PASS

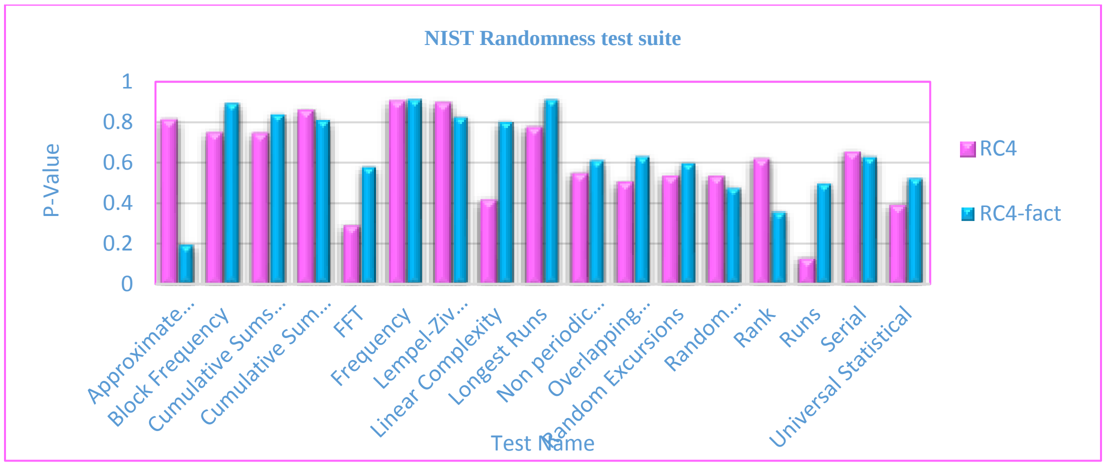

Figure. 2 NIST Statistical Test for RC4 and RC4-Fact.

8. THE INTRODUCED SINGLE BYTE BIAS ATTACK ALGORITHM

Isobe *et al.* [24] suggested efficient plaintext recovery attacks on RC4 algorithm that can retrieve all bytes of the plain text from the ciphertexts in the broadcast setting when the same plaintext is encrypted with different keys. AlFardan *et al.* [12] and Hammood *et al.* [25] at the same time, used the same concept and determined the plaintext recovery attacks and applied it on single-byte bias attack on TLS. Their attack successfully recovered the first 256 bytes of keystream with likelihood roughly 1 from 2^{24} ciphertexts encrypted with different random keys. This work determines a newly designed fast algorithm for calculating single byte bias attack on RC4 and retrieving the first 32 bytes of any plain text used, illustrated in the algorithm 5. The idea of this algorithm is based on the work of [12] and [25]. The concept of this algorithm aims to quarry the biases in the first 32 bytes of the RC4 keystream by finding the keystream value with the highest bias (K_i) in each position (i). The encryption of the same plaintext (P_i) with various random and independent keys generates many ciphertexts (C_i) that used to detect the most duplicated byte in each position to use it as the bias that is shifted as the value of the plain text. The most duplicate appearing bytes of the keystream are XOR-ed with the most duplicate appearing bytes of ciphertext to retrieve the plain text.

Algorithm 5. Single Byte Bias Attack

```
Input: Key [$k_1$, $k_2$, …., $k_{16}$], Plaintext $_x$.
Output: Plaintext*, Frequency of Plaintext*.
1.  For (X = 1 to N), where N = $2^{18}$, $2^{21}$, or $2^{24}$.
        1.1. For (n = 1 to $2^{34}$) Do
            1.1.1. x = 0, y = 0
            1.1.2. Call Algorithm 1: KSA
            1.1.3. Call Algorithm 2: PRGA
            1.1.4. Deducting new key with a length of 16 bytes from each generated key to be new secret key.
        1.2. For (col = 0 to key Length)
            1.2.1. For (row = 0 to $2^{34}$)
                   Set key [row] [col] as string
            1.2.2. For (x = 1 to values.Count)
                     If (values [x] = value)
                         i.   Increment count by 1
                         ii.  Key position = col
                         iii. Key value = value
                         iv.  Number of frequents = (count / ($2^{34}$ * 16))
2. Calculate Max-Frequent [Key sequence $_i$] of each position.
3. Ciphertext $_i$ = Encryption of Plaintext $_I$ with Key sequence $_i$
4. For (X = 1 to N).
        4.1. For (n = 1 to $2^{34}$) Do
            4.1.1. x = 0, y = 0
            4.1.2. Call Algorithm 1: KSA
            4.1.3. Call Algorithm 2: PRGA
            4.1.4. Deducting new key with a length of 16 bytes from each generated key to be new secret
                   key.
        4.2. For (col = 0 to key Length)
            4.2.1. For (row = 0 to $2^{34}$)
                   Set key [row] [col] as string
            4.2.2. For (x = 1 to values.Count)
                       If (values [x] = value)
                         i.   Increment count by 1
                         ii.  Key position = col
                         iii. Key value = value
                         iv.  Number of frequents = (count / ($2^{34}$ * 16))
5. Calculate Max-Frequent [Cipher text $_i$] of each position.
6. Plaintext*[X]= Encryption of Max-Frequent [Key-sequence $_i$] with Max-Frequent [Cipher text $_i$]
7. If Plaintext*[X] = Plaintext[X]
        Counter = Counter+1
```

8. Frequency of Plaintext * = (Counter * 100 /N)
9. Output: Plaintext *, Frequency of Plaintext *.

The execution time of single byte bias attack algorithm for RC4 is fast and requires about 1 second when using 218 keys, and for RC4-Fact was 3 seconds so the proposed algorithm is difficult for the cryptanalyzer. This algorithm successfully retrieved the first 32 bytes of RC4 plain text while RC4-Fact is robust to this attack and couldn't retrieve any of the plain text bytes. The recovery rate for RC4 and RC4-Fact is determined in Figures 3,4, and 5.

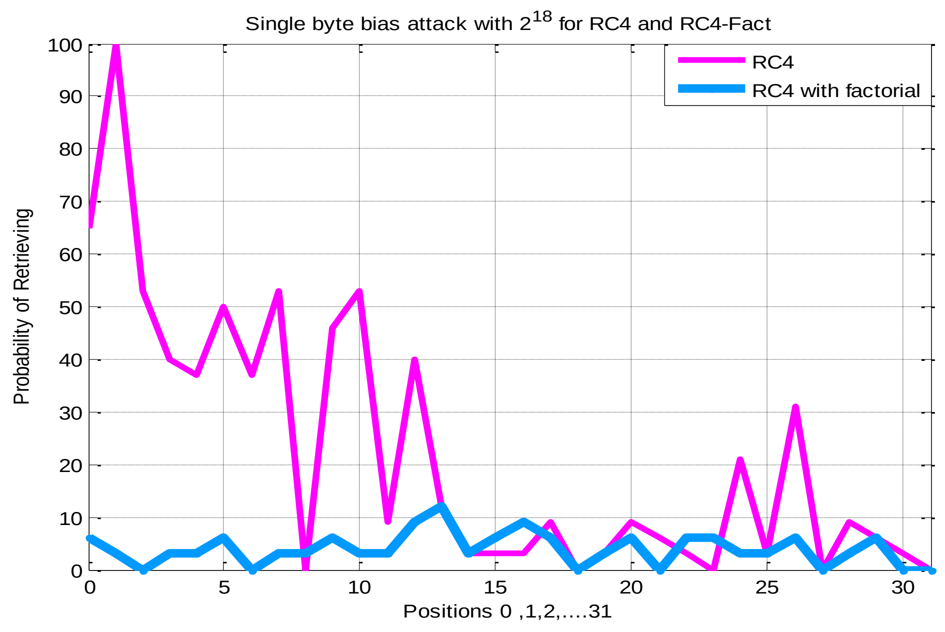

Figure. 3 The recovery rate for the first 32 position with 2^{18} for RC4 and RC4-Fact.

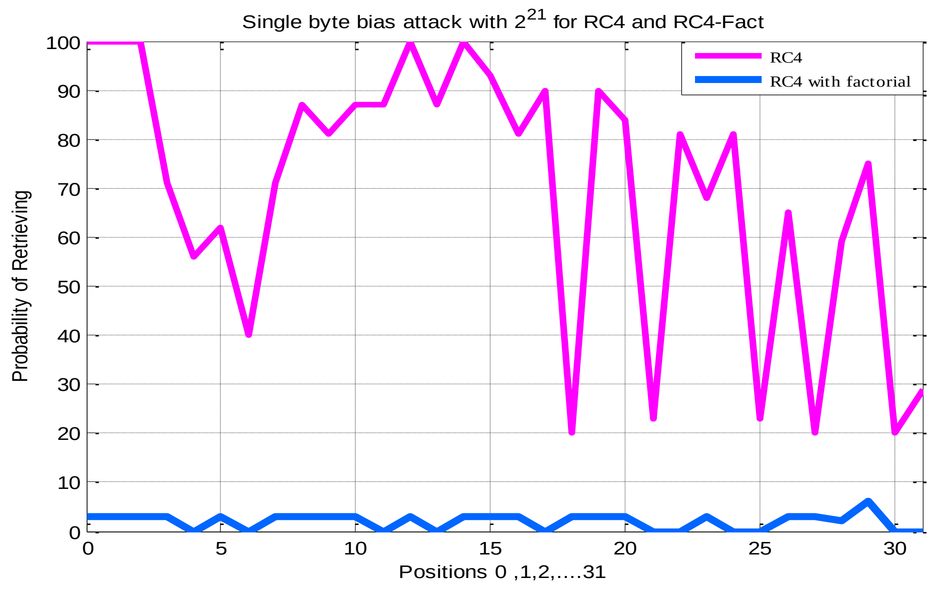

Figure. 4 The recovery rate for the first 32 position with 2^{21} for RC4 and RC4-Fact.

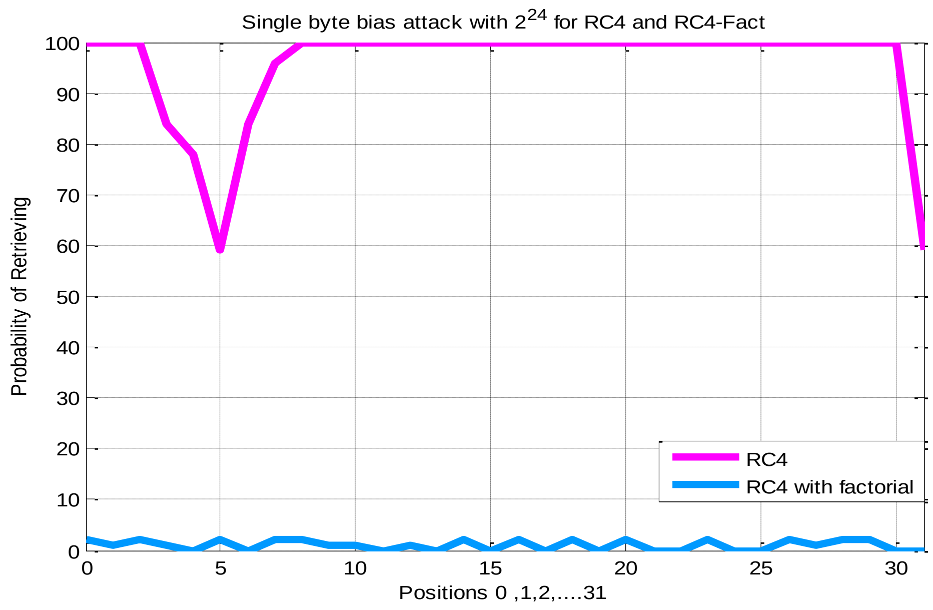

Figure. 5 The recovery rate for the first 32 position with 2^{24} for RC4 and RC4-Fact.

9. 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. It is simple and fast in implementation but it has weaknesses in its key stream bytes that these bytes are biased to some different values of the private key. RC4 biases are now quarried for making practical attacks on TLS. In this work, new algorithm is proposed, it 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. A new single byte bias attack algorithm is designed for attacking RC4 based on its single byte bias and retrieving all the first 32 bytes of RC4 plain text with the likelihood of 100% and the proposed algorithm is resistant to this attack. 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.

ACKNOWLEDGEMENT

Authors give many thanks to the College of Computer Sciences and Information Technology, University of Anbar, and to the Assist. Prof Dr. Maitham M. Hammood for providing support in order to carry out this research. Moreover, special thanks to the project supervisor Prof. Dr. Ali M. Sagheer for supporting us in this research and project.

REFERENCES

1. Mantin, I. and A. Shamir. *A practical attack on broadcast RC4*. in *International Workshop on Fast Software Encryption*. 2001: Springer.
2. Hammood, M.M., K. Yoshigoe, and A.M. Sagheer, *RC4-2S: RC4 stream cipher with two state tables*, in *Information Technology Convergence*. 2013, Springer. p. 13-20.
3. Sepehrdad, P., S. Vaudenay, and M. Vuagnoux. *Discovery and exploitation of new biases in RC4*. in *International Workshop on Selected Areas in Cryptography*. 2010: Springer.
4. McKague, M., *Design and analysis of RC4-like stream ciphers.* 2005.
5. Prasithsangaree, P. and P. Krishnamurthy. *Analysis of energy consumption of RC4 and AES algorithms in wireless LANs*. in *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*. 2003: IEEE.

6. Rahma, A.-M.S. and L.D.A.M. Sagheer, *DEVELOPMENT OF RC4 STREAM CIPHERS USING BOOLEAN FUNCTIONS.*
7. Stošić, L. and M. Bogdanović, *RC4 stream cipher and possible attacks on WEP.* Editorial Preface, 2012. **3**(3).
8. Maitra, S. and G. Paul. *New form of permutation bias and secret key leakage in keystream bytes of RC4*. in *International Workshop on Fast Software Encryption*. 2008: Springer.
9. Maitra, S. and G. Paul. *Analysis of RC4 and proposal of additional layers for better security margin*. in *International Conference on Cryptology in India*. 2008: Springer.
10. Garman, C., K.G. Paterson, and T. Van der Merwe. *Attacks only get better: Password recovery attacks against RC4 in TLS*. in *24th USENIX Security Symposium (USENIX Security 15)*. 2015.
11. Paul, S. and B. Preneel. *Analysis of non-fortuitous predictive states of the RC4 keystream generator*. in *International Conference on Cryptology in India*. 2003: Springer.
12. AlFardan, N., et al. *On the security of RC4 in TLS*. in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*. 2013.
13. Hammood, M.M. and K.Y.a.A.M. Sagheer, *Enhancing Security and Speed of RC4.* Int. J. Com. Net. Teach, 2015. **3**(2).
14. Khine, L.L., *A New Variant of RC4 Stream Cipher.* Assessment, 2009. **88**: p. 3138.
15. Bokhari, M., S. Alam, and F.S. Masoodi, *Cryptanalysis techniques for stream cipher: a survey.* International Journal of Computer Applications, 2012. **60**(9).
16. Wong, K.K.-H., 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-Volume 105*. 2010: Australian Computer Society, Inc.
17. Orumiehchiha, M.A., et al. *Cryptanalysis of RC4 (n, m) Stream Cipher*. in *Proceedings of the 6th International Conference on Security of Information and Networks*. 2013: ACM.
18. Mister, S. and S.E. Tavares. *Cryptanalysis of RC4-like Ciphers*. in *International Workshop on Selected Areas in Cryptography*. 1998: Springer.
19. Roos, A., *A class of weak keys in the RC4 stream cipher*. 1995, September.
20. Jindal, P. and B. Singh. *Performance analysis of modified RC4 encryption algorithm*. in *Recent Advances and Innovations in Engineering (ICRAIE), 2014*. 2014: IEEE.
21. Billet, M.R.O., *New stream cipher designs*. 2008: Springer.
22. Pardeep and K. Pushpendra, *PC1-RC4 and PC2-RC4 algorithms: Pragmatic enrichment algorithms to enhance RC4 stream cipher algorithm.* International Journal of Computer Science and Network (IJCSN), 2012. **1**(3): p. 98-108.
23. Fluhrer, S.R. and D.A. McGrew. *Statistical analysis of the alleged RC4 keystream generator*. in *International Workshop on Fast Software Encryption*. 2000: Springer.
24. Isobe, T., et al. *Full plaintext recovery attack on broadcast RC4*. in *International Workshop on Fast Software Encryption*. 2013: Springer.
25. Hammood, M.M. and K. Yoshigoe. *Previously overlooked bias signatures for RC4*. in *2016 4th International Symposium on Digital Forensic and Security (ISDFS)*. 2016: IEEE.

AUTHOR PROFILE

Ali M. Sagheer is a Professor in the Computer College at Al-Anbar University. He received his B.Sc. of 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 conferences and scientific journals.

Sura M. Searan has received her B.Sc. of 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.

Rawan A. Alsharida is an assistant lecturer in the department of computer science at University of Tikrit, she received Master of Information Science, information Quality in (2015), University of Arkansas at Little Rock, USA. From 2005 she was Bachelor of Computer Science, University of Tikrit, Tikrit, Iraq.