Shaping the Future of ICT: Trends in Information Technology, Communications Engineering, and Management

Ibrahiem M. M. El Emary, Anna Brzozowska

Hardback
£140.00

July 14, 2017 by CRC Press

Reference - 520 Pages - 225 B/W Illustrations

ISBN 9781498781183 - CAT# K29959

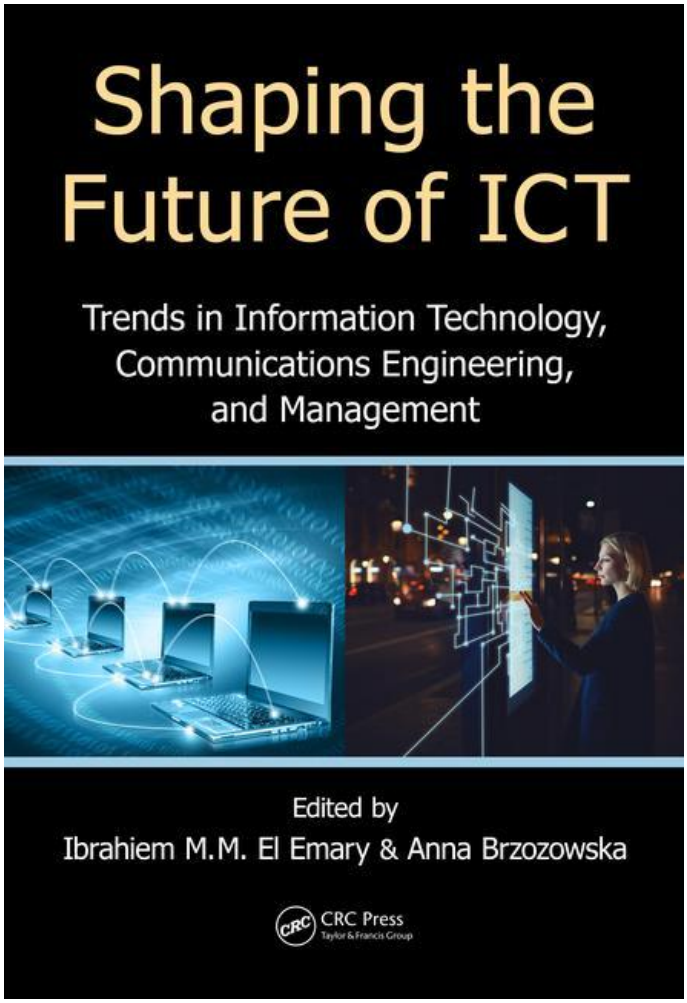

Table of Contents

Section II COMMUNICATION SYSTEMS

Design of New Algorithms to Analyze RC4 Cipher Based on Its Biases

Ali M. Sagheer
University of Anbar, Iraq

Sura M. Searan
University of Anbar, Iraq

Abstract

RC4 is an effective stream cipher and it is more popular. It is fast, simple and suitable for software and hardware. It is used in many applications, it was analyzed by different researchers and several weaknesses were detected, these weaknesses 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. Analyzing these bytes makes them probable for attacking RC4. This chapter shows the analysis of RC4 key stream based on its single and double byte biases by using new designed algorithms that calculate the bias in a standard time (few seconds for single bias and few minutes for double bias). Also, shows the single byte bias attack on RC4 by using the newly designed algorithm. The results showed that the bias of RC4 keystream proved and got the same results that shown previously with less time 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%. The analysis of 256 positions required additional requirements such as supercomputer and message passing interface (MPI) environment that not available in Iraq, therefore; the analysis is done for 32 positions to reduce the search space.

Keywords

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

1. Introduction

Information security is a process that an organization protects and secures its systems [1]. Information can be protected by encrypting it using one of encryption algorithms. Many factors needed to take into accounts such as security, the characteristics of the algorithm, time complexity and space complexity. The main objective of the cryptography is not only used to provide privacy but also to provide solutions to other problems such as integrity, authentication, non-repudiation,

and availability [2]. There are many encryption algorithms that widely used in wired networks. In symmetric encryption, when the key size is small, it must be very efficient and encryption time can be quicker. Various encryption ways that used in wireless devices based on symmetric encryption, such as RC4 algorithm [3]. RC4 stream cipher is an effective and the most popular algorithm. It is one of important encryption algorithms. It's designed by Ron Rivest in 1987 and it is called "Ron's Code 4". It's based on the use of random permutation [4] and was a trade confidential to 1994. It is used in commercial software packages such as MS Office, Oracle Secure SQL, and used in network protocols such as IP Sec, Wired Equivalent Privacy (WEP) Protocol [5] and used to protect wireless networks as part of Wi-Fi Protected Access (WPA) protocols and to protect the internet traffic as part of Secure Socket Layer (SSL) protocol and Transport Layer Security (TLS) protocol [6]. RC4 was analyzed by different people and different weaknesses were detected. [7]. The attack on this algorithm described by Mantin and Shamir [8] and Fluhrer [9]. The main contributions of this work are designing new efficient and fast algorithms to analyze the RC4 algorithm based on single byte bias and double byte bias. Also, designing an algorithm for a single byte bias attack that can retrieve all the first 32 bytes of the plain text of RC4 with probability of 100% in a few seconds of time.

2. Literature Survey

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 that 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 significant to discriminant 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 [8]. Fluhrer S.R. & McGrew D.A. (2001) the first researchers that determined a new method to distinguish 8-bit of RC4 from random bits and discovered a new type of bias described as double byte bias in a consecutive pair of bytes. They discovered long-term biases for RC4, ten conditions stated the positive biases that mean their likelihood are higher than the meant value and two conditions stated negative biases that mean

their likelihood is lower than the intended value [9]. 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 2^{44} random keys [20]. Hammood M. M. and Yoshigoe K. (2016) determined different biases in the RC4 keystream and analyzing developed algorithms that determined in [6] by using C programming and Message Passing Interface environment and experiments are executed by using a high execution system with 256 processor units [21]. This work implemented the proposed algorithms on a personal computer and got the same biases that shown previously and in less time (seconds only).

3. RC4 Algorithm Description

Many of stream cipher algorithms 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 [10]. 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 implemented sequentially [11]. 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. In KSA, can produce initial Permutation of RC4 by scramble the corresponding permutation using the key. This permutation (State) in KSA used as an input to the second step (PRGA) that generates the final key stream [11]. RC4 starts with the permutation and use 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 which 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:

3.1 Key Scheduling Algorithm (KSA)

The key-scheduling algorithm starts with the permutation of the state table (State [i]). This algorithm consists of two steps: The initialization step that the State[i] is set to the identity permutation which is processed for 256 iterations and the other is mixing step, it uses a key with N bytes to continue swapping values of the state for generating new key stream dependent permutations [2]. This portion of RC4 equips the state table that used in pseudo random generation algorithm to generate the final key [16].

Algorithm 1. KSA of RC4 Algorithm
Input: Key. **Output:** State[i].
1. For (i = 0 to 255) 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[i].

3.2 Pseudo-Random Generation Algorithm (PRGA)

The state table that got from KSA is swapped with itself by using a known indicator and random indicator. A random index is produced consecutively by using the same values from the prior iteration. Then, the state table swapped by using these values. The output is generated by taking the modular addition for the values at index pointers [11]. This algorithm is continuously mixed the permutation that stored in the state and select different values from the state permutation to be as an output. This algorithm started corresponds to n-bit word as the key stream with set i and j to 0 and loops through four simple operations that increment (i) as a counter and increment

(j) pseudo-randomly swapped the two values of S-box (State[i] and State[j]) and output the value of S-box pointed to by (State[i] + State[j]) [7].

Algorithm 2. PRGA of RC4 Algorithm

Input: State[i], Plaintext n.

Output: Key sequence.

1. i = 0, j = 0.

2. While generating output:

    2.1. i = (i + 1) mod 256

    2.2. j = (j + State[i]) mod 256

    2.3. Swap (State[i], State[j])

    2.4. K sequence = State [State[i] + State[j]] mod 256

3. Output: Key sequence.

Cipher text n = Key sequence n $\oplus$ Plaintext n [14]

4. RC4 Algorithm Weaknesses

There are several weaknesses found in RC4 algorithm. Some of these are easy and can be resolved, but other weaknesses are dangers that can be quarried by the attackers. RC4 failed in providing a high level of security because of the biases in the bytes of the keystream [15]. Roos [16] found RC4 weaknesses that 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. So 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 [16]. Mantin and Shamir [8] found the main weakness of RC4 in the second round. The likelihood of zero output bytes. Fluher [9] found a large weakness if anyone knows the private key portion

then potential to attack fully over the RC4 [9]. Paul and Maitra [17] 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 [18]. So 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 [19]. The attack aims to retrieve the main key, the internal state, or the final key stream to access to the original messages [23].

5. The Proposed Single Byte Bias Algorithm to Analyze RC4 Cipher.

The first researchers that denoted the bias in the key stream of RC4 were mantin and Shamir after various researchers studied different biases. In this chapter, RC4 and developed algorithms were implemented in C# programming and the new efficient experiment was designed based on the idea of [20] and [21] for proving single bias in the first 32 bytes of the keystream which is summarized as algorithm 3. The RC4 output has shown the same biases that described in the previous researches. The experiment executed with generated key stream ranging from 2^{20} to 2^{34} with an independent secret key size of random 16 bytes. The frequents calculated by the following equation:

$$\text{Frequents}=2^{34}/\text{State-length}. \quad (1)$$

When the likelihood of the frequents of any value is higher than the average, this taken as positive bias and when the probability less than the average, this operates as negative bias.

$$\text{Av.}=\text{Frequents}/2^{34}. \quad (2)$$

Algorithm 3. Measuring Distributions of Bytes of Keystream
Input: Key [k_1, k_2, …, k_{16}]. **Output:** Key position (Kp), key value (Kv), and frequents number in each position (Kf).
1. For (n = 1 to 2^{34}) Do 1.1. x = 0, y = 0 1.2. Call Algorithm 2.1: KSA

1.3. Call Algorithm 2.2: PRGA

1.4. Deducting new key with a length of 16 bytes from each generated key to be new secret key.

2. For (col = 0 to key Length)

2.1. For (row = 0 to 2^{34})

2.1.1. Set key [row] [col] as string

2.2. For (x = 1 to values. Count)

2.2.1. If (values [x] = 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 frequents = (count / (2^{34} * 16))

Output: Kp, Kv, and Kf for each position of key stream bytes

Different biases in the short-term key stream of RC4 were identified previously. This work successfully regenerated these keystream byte probabilities for the first 32 positions.

Figures 1, 2, 3, and 4 represents the distribution of key stream bytes in the positions 1, 2, 16, and 32 respectively.

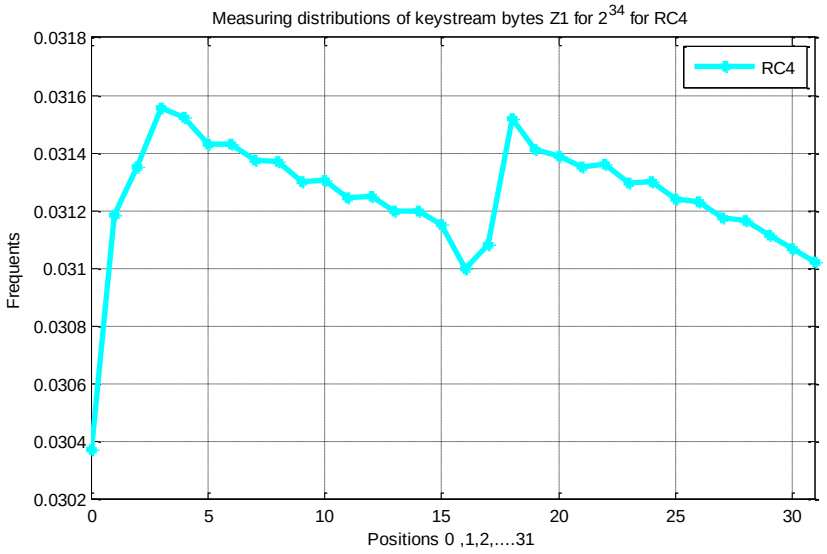

Figure 1. Distribution of Key Stream Bytes in the 1st Position with 2^{34} Secret Keys.

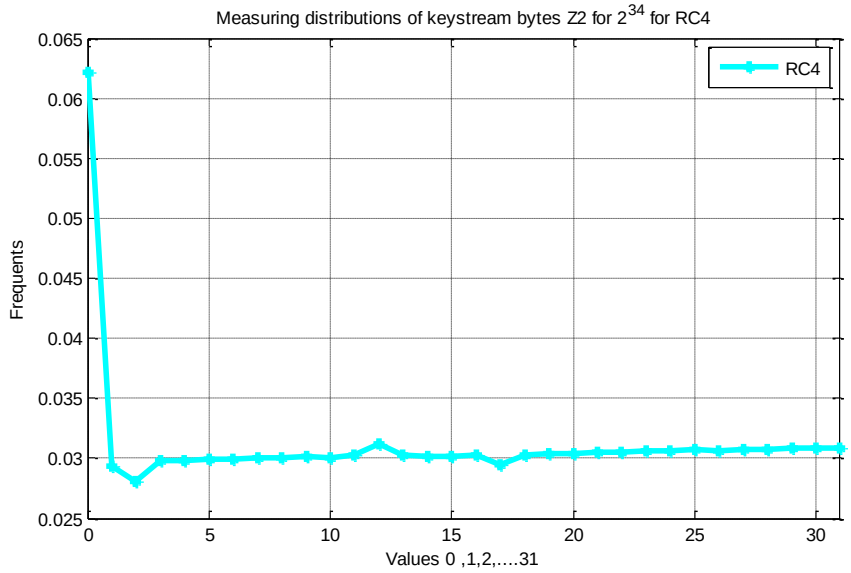

Figure 2. Distribution of Key Stream Bytes in the 2nd Position with 2^{34} Secret Keys.

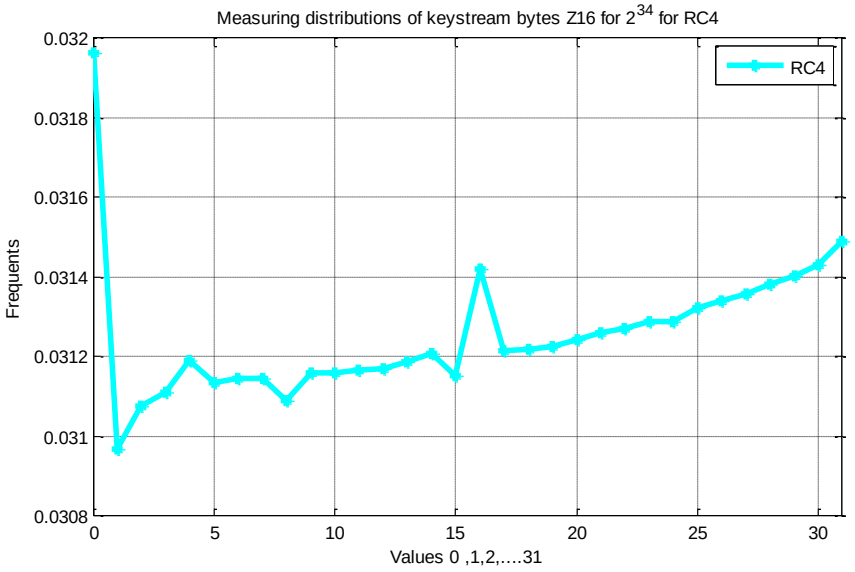

Figure 3. Distribution of Key Stream Bytes in the 16th Position with 2^{34} Secret Keys.

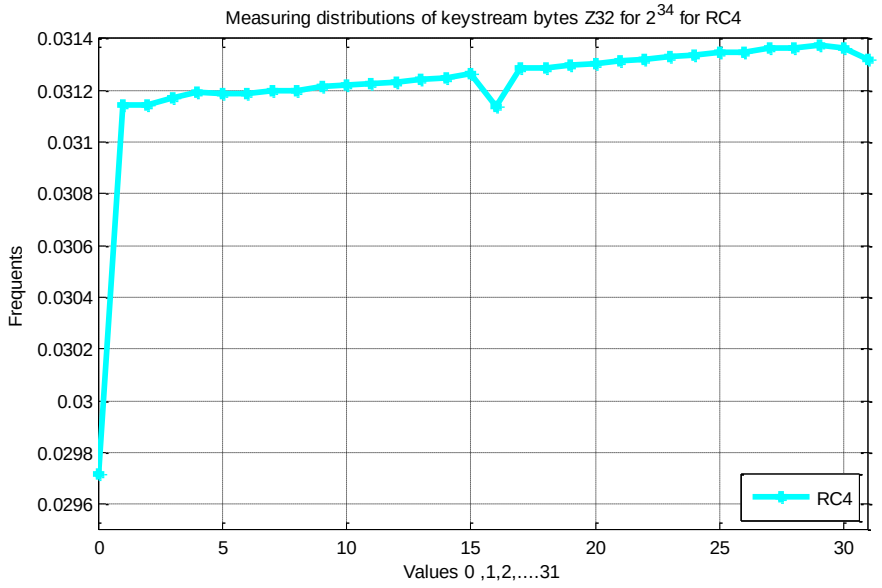

Figure 4. Distribution of Key Stream Bytes in the 32nd Position with 2^{34} Secret Keys.

Consider Pr ($K_i = n$), where i = 1, 2, …, N represented the round number in PRGA phase of RC4, and n = 0, 1, 2, …, N-1 represented the output keystream values as shown in figure 5. The spikes and apparent vertical walls through the figure represented the short-term bias in the first 32 positions of RC4 keystream. Particularly, the downward spike and the vertical wall in the front right represented the distributions of key stream bytes for K_1.

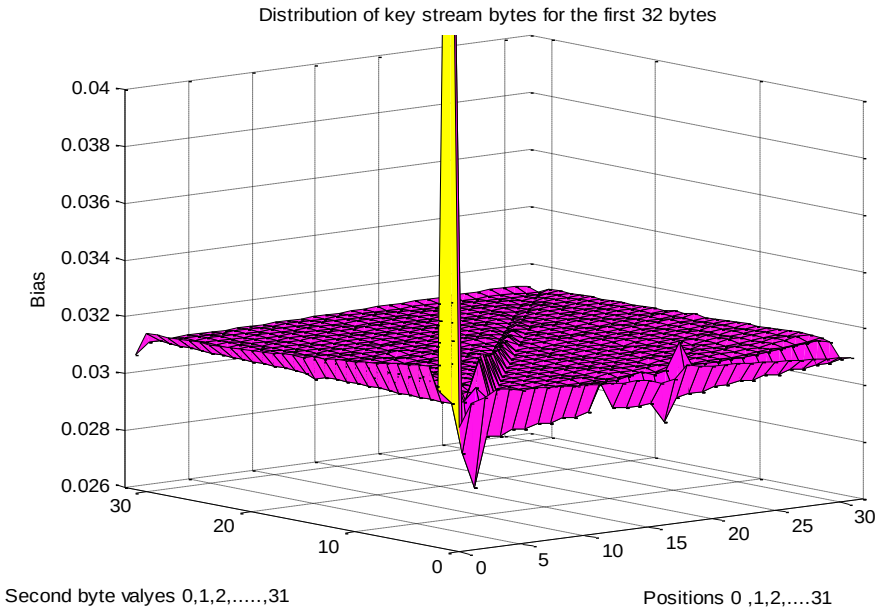

Figure 5. Measuring Distributions of RC4 Keystream for the First 32 Bytes with 2^{34}.

6. The Proposed Double Byte Bias Algorithm to Analyze RC4 Cipher.

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 resistant from the single-byte bias attack. However, the researchers have studied and were investigated for biases beyond initial bytes and different multi-byte biases have been discovered in the key stream of RC4. Fluhrer and McGrew [9] were the first researchers that discovered the biases in a consecutive pair of bytes (K_i, K_{i+1}) and detect long-term biases of RC4. They discover ten positive biases that mean their probability were higher than the desired value and detected two negative biases that mean their probability were lower the desired value. Hammood et al. [21] estimated the probability of the cipher for generating each pair of byte values though 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 endorse them work by AlFardan et al. They found two new positive biases not mentioned in [9] by Fluhrer and McGrew. In this chapter, the Fluhrer and McGrew biases and hammood bias is reproduced with 1024 keys of 16 bytes to generate 2^{32} keystream bytes after discarding the first 1024 bytes. Every key from these 1024 keys produces 2^{32}; therefore, the whole amount of the generated keys is 2^{42}. Algorithm 4 below designed to determine the measuring of double byte bias in a few seconds. The main idea of this algorithm is to measure the appearance of the consecutive pair (Z_i, Z_{i+1}) in each position of RC4 output.

Algorithm 4. Measuring Distributions of Key Stream Bytes (K_i, K_{i+1})

Input: K [k_1, k_2, …., k_{16}].

Output: Frequents of (K_i, K_{i+1}).

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 32
     - 2.2.2. j= (j + State[i]) mod 32
     - 2.2.3. Swap (State[i], State[j])
     - 2.2.4. Generated Key = State[(State[i] + State[j]) mod 32]

2.2.5. A[k][Generated Key][i1] = A[k][Generated Key][i1] +1

2.2.6. Deducting new key with 16 bytes to be new secret key.

2.2.7. k = Generated Key

2.2.8. i1 = (i1 + 1) mod 32

3. Output: A[k][Generated Key][i1].

Figures 6, 7, 8, and 9 show the distribution of (Z_r, Z_{r+1}) where Zr=0 and Zr+1=0, where Zr=30 and Zr+1=31, where Zr=31 and Zr+1=30, and where Zr=31 and Zr+1=31. Sequentially.

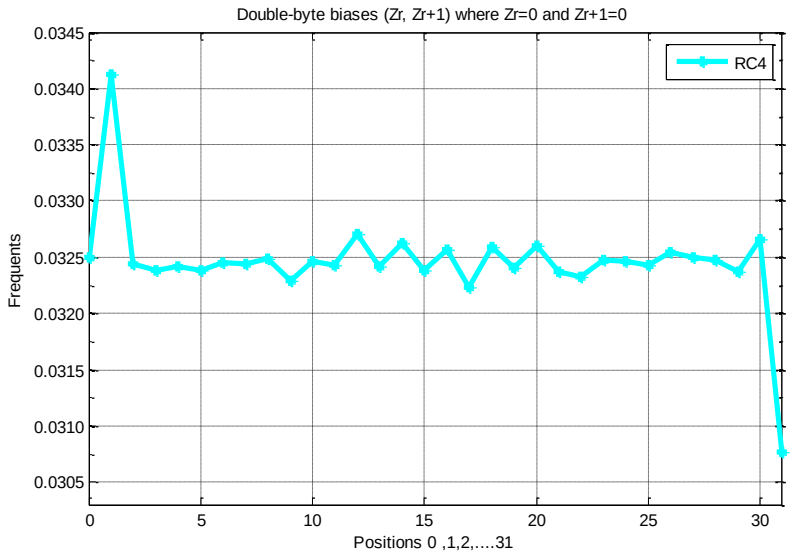

Figure 6. Double-Byte Biases (Z_r, Z_{r+1}) where Zr=0 and Zr+1=0.

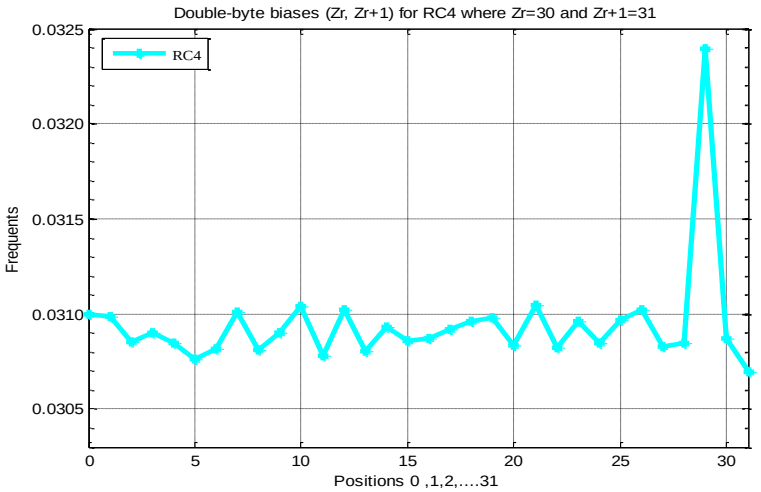

Figure 7. Double-Byte Biases (Z_r, Z_{r+1}) where Zr=30 and Zr+1=31.

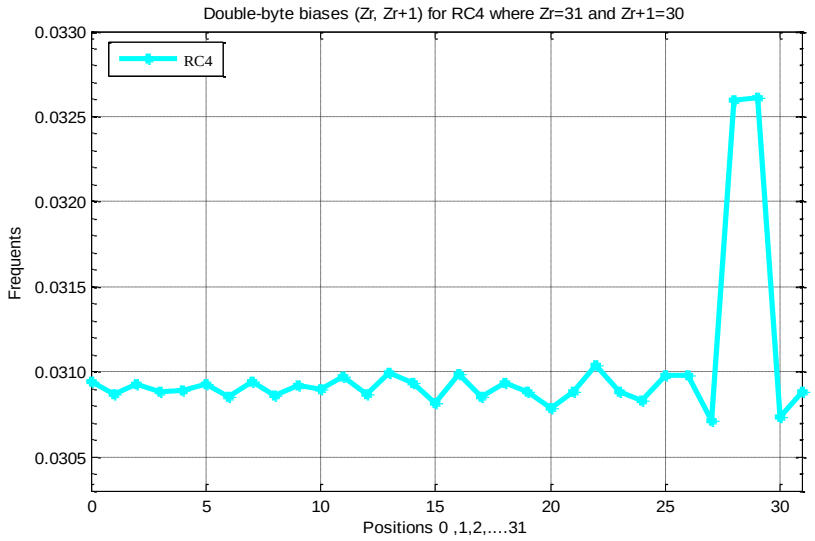

Figure 8. Double-Byte Biases (Z_r, Z_{r+1}) where Zr=31 and Zr+1=30.

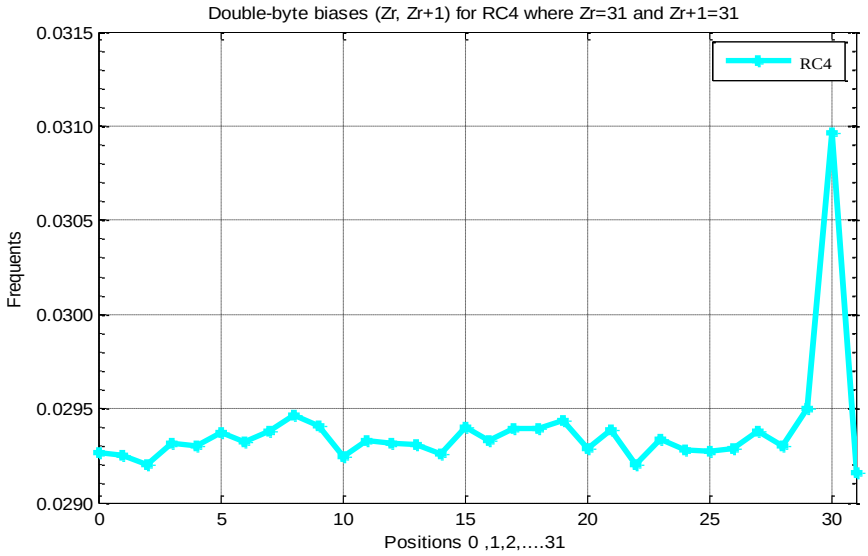

Figure 9. Double-Byte Biases (Z_r, Z_{r+1}) where Zr=31 and Zr+1=31.

Figures 10,11, and 12 show the results for running above algorithm that measures the distributions of keystream bytes (Z_i, Z_{i+1}) to discover possible double-byte biases for RC4 in the first 32 bytes.

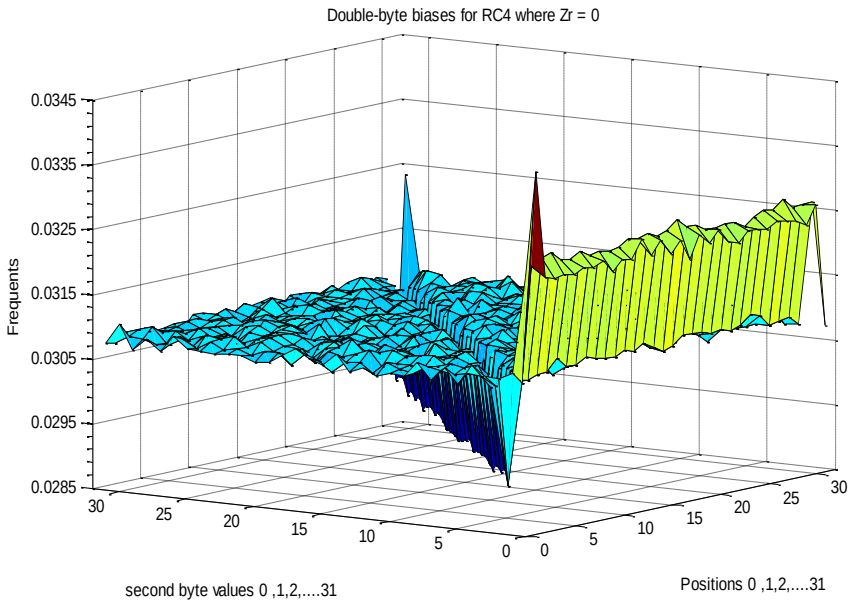

Figure 10. Double-Byte Biases (Z_r, Z_{r+1}) for RC4 where Z_r=0.

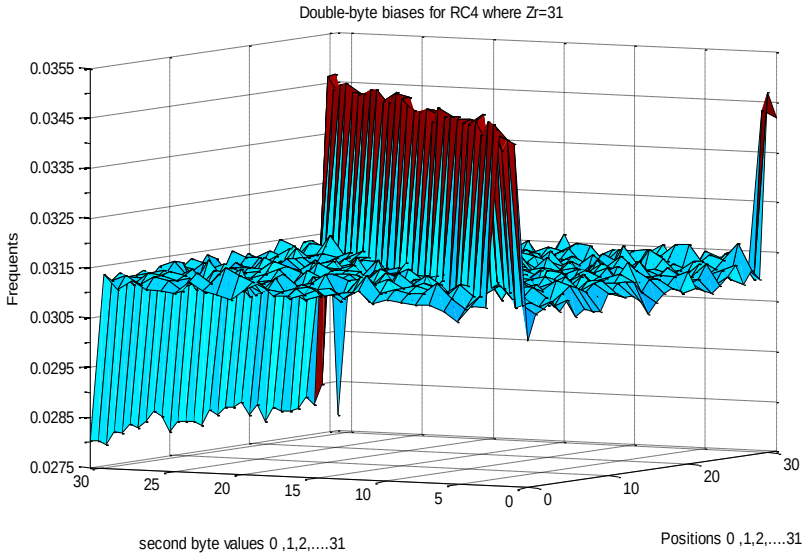

Figure 11. Double-Byte Biases (Z_r, Z_{r+1}) for RC4 where Z_r=31.

The figure below shows the distribution of (Z_r, Z_{r+1}) for all the first 32 bytes, where Z_r = i and Z_{r+1} = i for RC4.

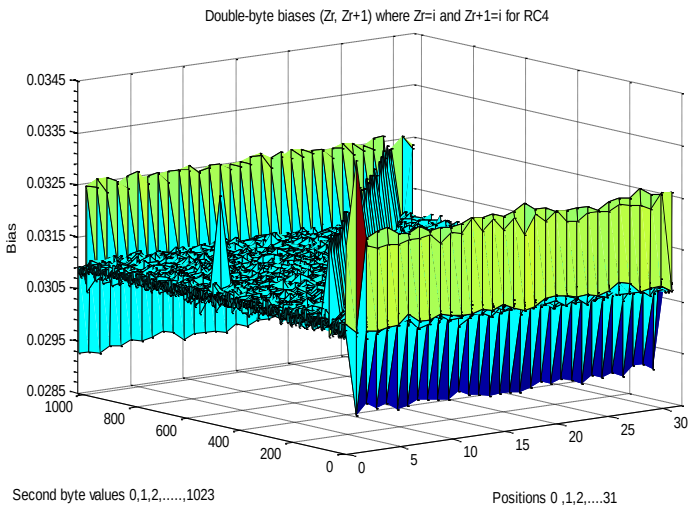

Figure 12. Double-Byte Biases (Z_r, Z_{r+1}) for RC4 where Z_r=i and Z_{r+1}=i.

7. The Proposed Single Byte Bias Attack Algorithm on RC4.

Isobe *et al.* [22] 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.* [20] and Hammood *et al*. [21] 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 chapter 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 [20] and [21]. 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 shifted as the value of the plain text. The most duplicate appearing bytes of the keystream is 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 $_i$.
Output: Plaintext*, Frequency of Plaintext*.

1. For (X = 1 to N), where N = 2^{18}, 2^{21}, or 2^{24}.
    1.1. Call algorithm 1: Measuring distributions of RC4 Keystream $_i$. bytes
    1.2. Calculate Max-Frequent [Key sequence $_i$] of each position.
    1.3. Ciphertext $_i$ = Encryption of Plaintext $_I$ with Key sequence $_i$
    1.4. Call algorithm 1: Measuring distributions of RC4 Ciphertext $_i$. bytes
    1.5. Calculate Max-Frequent [Cipher text $_i$] of each position.
2. Plaintext*[X]= Encryption of Max-Frequent [Key-sequence $_i$] with Max-Frequent [Cipher text $_i$]
3. If Plaintext*[X] = Plaintext[X]
    3.1. Counter = Counter+1
4. Frequency of Plaintext * = (Counter * 100 /N)
5. Output: Plaintext *, Frequency of Plaintext *.

The execution time of single byte bias attack is determined below in figure 13:

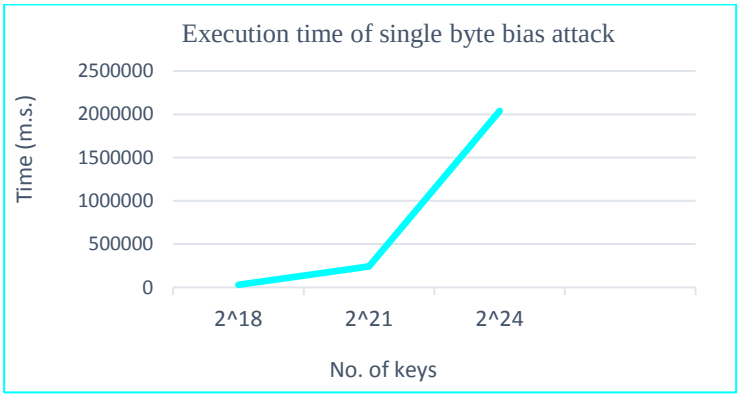

Figure 13. Execution Time of Single Bias Attack.

Figures 14, 15, and 16 shows the probability of retrieving the plain text bytes:

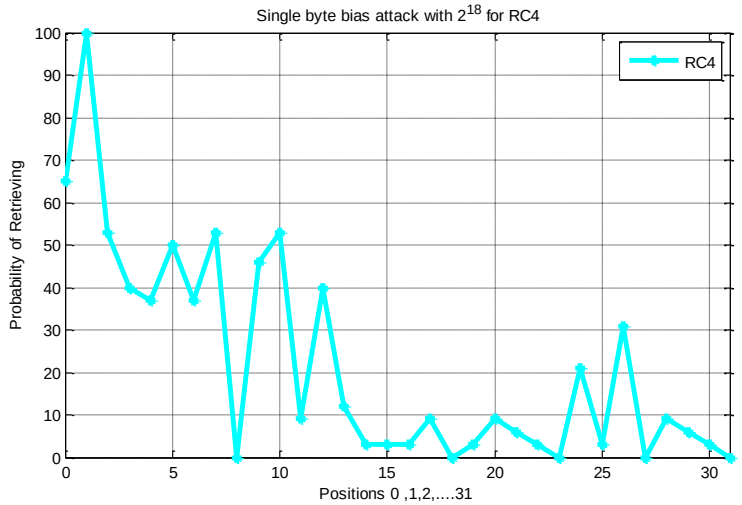

Figure 14. The Recovery Rate for the First 32 Position with 2^{18}.

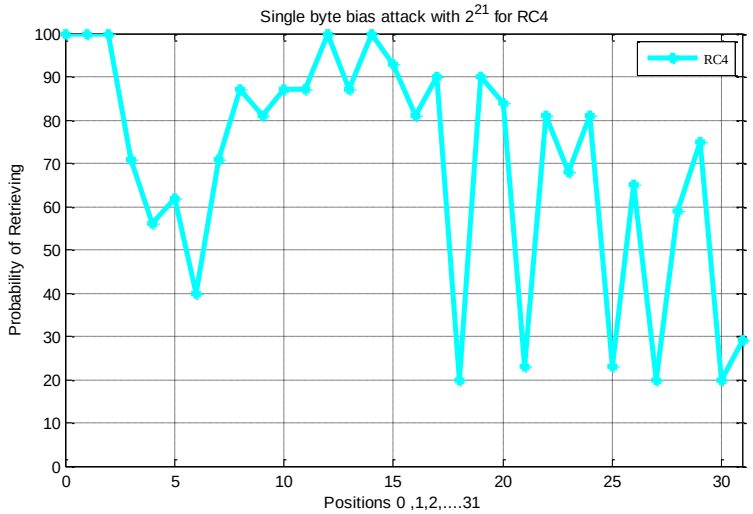

Figure 15. The Recovery Rate for the First 32 Position with 2^{21}.

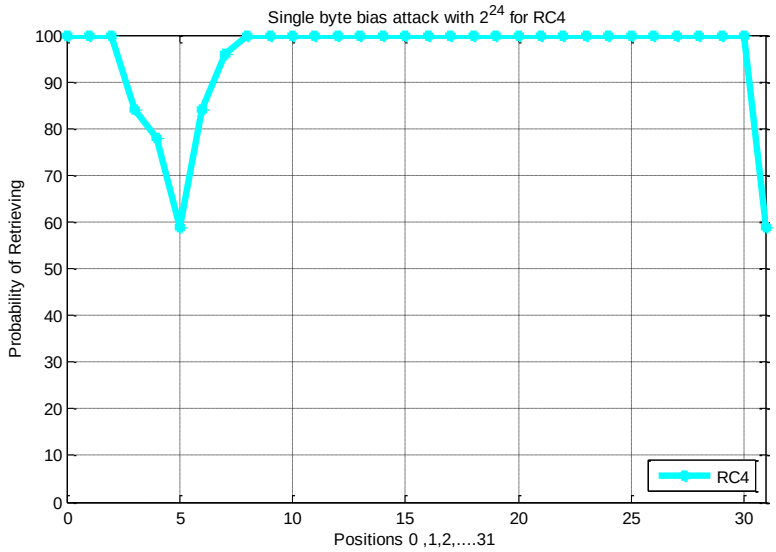

Figure 16. The Recovery Rate for the First 32 Position with 2^{24}.

Conclusions

RC4 is an important encryption algorithm that can be used for information protection on many communication networks as it 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 protocol. In this chapter, the analysis of RC4 algorithm is done for the first 32 positions by using newly designed fast algorithms and shown the same bias that shown previously. Also, 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%. As a future work, the proposed algorithms may be applied on 256 bytes by using parallel processors.

References

[1] Robshaw, M. & Billet, O. (2008). New Stream Cipher Designs: The eSTREAM Finalists. Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 4986, 1-6.

[2] Gupta, S. S. (2013). Analysis and Implementation of RC4 Stream Cipher (Doctoral dissertation), Indian Statistical Institute Kolkata, Kolkata, West Bengal, India.

[3] Prasithsangaree, P. & Krishnamurthy, P. (2003, December). Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs. In *Proceedings of Global Telecommunications Conference,* IEEE, 1443 (3), 1445-1449.

[4] Karahan, M. (2015). New Attacks on RC4A and VMPC. (Doctoral dissertation), bilkent university.

[5] Paul, G. (2007). Structural Weakness of the Key Scheduling of RC4. Jadavpur University: IFW 2007.

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

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

[8] Mantin, I. & Shamir, A. (2001). A Practical Attack on Broadcast RC4. *In Fast Software Encryption*, Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer-VerlagBerlin Heidelberg, 2355, 152-164.

[9] Fluhrer, S. R. & McGrew, D. A. (2001). Statistical Analysis of the Alleged RC4 Keystream Generator. Lecture Notes in Computer Science, Springer- Berlin Heidelberg, 1978, 19-30.

[10] Hammood, M. M., Yoshigoe, K., & Sagheer, A. M. (2013). 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. Springer Netherlands.

[11] Garman, C., Paterson, K. G., & Van der Merwe, T. (2015). Attacks Only Get Better: Password Recovery Attacks Against RC4 in TLS: In Presented as part of The 24th USENIX Security Symposium, 15, 113-128.

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

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

[14] Sivasankari, N. & Yogananth, A. (2014). Effective and Efficient Optimization in RC4 Stream. International Journal of Scientific Engineering and Technology, 3(6), 826-829.

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

[16] Roos, A. (1995). A class of weak keys in the RC4 stream cipher: South African, Vironix Software Laboratories: Westville.

[17] Maitra, S. & Paul, G. (2008). New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4. Lecture Notes in Computer Science, in Fast Software Encryption, 5086, 253-269.

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

[19] Ohigashi, T., Isobe, T., Watanabe, Y., & Morii, M. (2013). How to Recover Any Byte of Plaintext on RC4. Lecture Notes in Computer Science (8282), 155-173.

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

[21] Hammood, M. M., Yoshigoe, K., & Sagheer, A. M. (2015). Enhancing Security and Speed of RC4. International Journal of Computing and Network Technology, 3(2), 2210-1519.

[22] Isobe, T., Ohigashi, T., Watanabe, Y., & Morii, M. (2014). Full Plaintext Recovery Attack on Broadcast RC4. *Lecture Notes in Computer Science*, 8424, 179-204.

[23] Sagheer, A. M., Searan, S. M., Alsharida, R. A. (2016). Modification of RC4 Algorithm to Increase its Security by Using Mathematical Operations. Journal of Software Engineering & Intelligent Systems, 1(2).