A Proposed Modification on RC4 Algorithm by Increasing its Randomness

Assist. Prof. Dr. Soukaena H. Hashem
soukeana.hassan@yahoo.com
University of Technology - Computer Science Department
Ayman B. Jasim
ayman_bashar2000@yahoo.com
University of Technology - Computer Science Department

Abstract: *Wired Equivalent Privacy (WEP) protocol was adopted as security protocol to protect IEEE 802.11 Wireless LAN from unauthorized access, eavesdropping and other attacks. Over the past few years, several serious security flaws discovered in WEP protocol and its underlying cryptographic primitives (data integrity and encryption algorithms). These flaws lead to a number of practical attacks that demonstrate that WEP fails to achieve its security goals. In this research an attempt is accomplish to improve the encryption algorithm "Standard RC4" of WEP protocol through proposing an enhanced algorithm named as "Proposed RC4+S" that would overcome the security flaws and strengthen the level of security and protection for WEP protocol. Then evaluate the proposed algorithm "RC4+S" to prove the advance of research proposal, which illustrated through the following result: the proposed RC4+S algorithm increases (secret-key) the randomness by approximately more than (20%), thus leading to improve output*

(ciphertext) randomness of modified WEP protocol compared to standard protocol.

Keywords: Wired Equivalent Privacy (WEP) protocol, IEEE 802.11Wireless LAN, Eavesdropping, Cryptographic primitives, RC4 Algorithm.

1. Introduction

Wireless networks (also called IEEE 802.11 WLAN) are becoming more and more popular today "Cisco has predicted that the number of wireless connected devices in 2014 is going to surpass the world's population". Major companies are using wireless network as integral part of their business environments. The main reason for wireless network popularity that enables users to access local resources no matter of their location. Wireless network uses electromagnetic (radio) waves to connect devices and transmit information without using cables [1]. Most wireless networks are based on the IEEE 802.11 standards, where the formulation of such wireless networking system is governed by IEEE 802.11 standards that manage network usage as well as security mechanism to be performed over the vulnerable wireless medium [2].

WEP protocol is the basic part of IEEE 802.11 standards for the protection of wireless networks. The primary goal of WEP protocol is to provide secure data transmission over wireless networks in the same way as it is in the wired networks during transmission. Regardless of the underlying transmission techniques of wireless networks, IEEE 802.11 defines WEP protocol in two stages [3,4]: (A) data integrity algorithm using CRC-32, (B) Encryption algorithm using RC4.

Standard WEP protocol contains some security flaws that give an incentive to number of attacks such as, the famous FMS attack by Fluhrer et al. that shows the use of RC4 algorithm to generate random keystream for encryption is subject to weaknesses [5]. Pyshkin et al., have demonstrated an active attack on the WEP

protocol that is able to recover a 104-bit WEP key using less than 40,000 frames with a success probability of 50%, and 85,000 packets are needed to achieve success by 95% probability of successful execution [6]. Development did not stop after the FMS attack was published; instead people started looking for more correlations between the secret root key and the first bytes of output of RC4-PRGA. A person under the name "KoreK" posted an implementation of a WEP cracker known as "KoreK key recovery attack" [7].

This research paper aims to increase the degree of secret-key randomness of encryption algorithm "standard RC4" for WEP protocol through proposing an enhanced algorithm named as "Proposed RC4+S" that would overcome the security flaws and strengthen the level of security and protection for WEP protocol.

2. Related Work

- Pardeep et al., in 2012 proposed a pardeep cipher called (PC-RC4) which is adding a new improvements and extension to RC4 algorithm. In the PC-RC4, randomness in KSA and PRGA is improved to make it stronger, but increase the time of algorithm execution [8].

- Jagdeep Singh et al., in 2013 they proposed an algorithm providing two stages encryption and decryption called (Robust-RC4). Where through encryption and decryption, the data is being processed two times. The proposed algorithm is trying to make RC4 algorithm too strong. Also proposed some enhancement inside the KSA and PRGA sub steps. [9].

- Subhamoy Maitra1 et al., in 2013 proposed a ($RC4^{+}$) algorithm which is a modified version of standard RC4 with three-layer architecture of the permutation phase after the first initialization phase. That removes many weaknesses of the KSA and performs addition permutation steps in the

state-array for each output byte in PRGA to strengthen and randomize the output cipherkey [10].

- Razi Hosseinkhani et al., in 2012 they describe how generate dynamic S-Box that is changed with every changing of cipher key, which increasing the cryptographic strength of algorithm. The S-Box component that used in AES is fixed, and not changeable. Where if we were able to generate this S-Box dynamically, we will increase the cryptographic strength of AES cipher [11].

3. RC4 Algorithm

RC4 is a stream cipher invented in 1987 by "Ron Rivest", it is a variable keysize with byte-oriented operations. As a result of its simplicity and speed in software, RC4 is considered widely used stream cipher as encryption algorithm in popular internet protocols such as WEP, WAP and SSL/TLS.

RC4 algorithm work is based on random permutation of state array using input-key (seed), that state array is used to generate a pseudo-random keystream used for encryption by XORing it with the plaintext, decryption is performed in the same way. RC4 algorithm is called "Pseudo-random generator", because it generates a sequence of numbers that are close to the characteristics of true-random numbers, but are not truly random [12].

RC4 keystream is generated from a "variable length" input-key (seed) using an internal state which consists of the following [12]:

- An array of 256-bytes symbolized as "S", including the permutation of all 256 possible bytes.
- Two indexes of 8-bit symbolized as "i" and "j", used to point the position of elements in the "S" array.

The seed-key is used to initialize the "S" array of internal state, that phase is known as the " Key Schedule Algorithm (KSA)", then the initialized "S" array is used to generate a pseudo random

sequence of keystream bytes, that phase is known as "Pseudo-Random Generation Algorithm (PRGA)" [12].

3.1 Key-Scheduling Algorithm (KSA):

As mentioned previously, the KSA is used for "initializing" and "permutation" of the "S" array. The first step is initializing the "S" array with the equivalent permutation (the values in the array are equal to their index). Once the "S" array is initialized, the next step is shuffling the "S" array using seed-key to generate the "S" array, where "S" is processed for 256 iterations to make it a permutation array [12], Algorithm (1) introduces a pseudo-code corresponding to KSA.

Algorithm (1): Pseudo-code Corresponding to KSA

```
Input: Seed key
Output: Initialize and permutation S-array
Process:
For i from 0 to 255
   S[i] := i
Endfor
j := 0
For i from 0 to 255
   j := (j + S[i] + key[i mod keylength]) mod 256
   swap values of S[i] and S[j]
Endfor
```

When "i" index reaches 256, that means the iteration is complete and the "S" array has been properly initialized, see Figure (1).

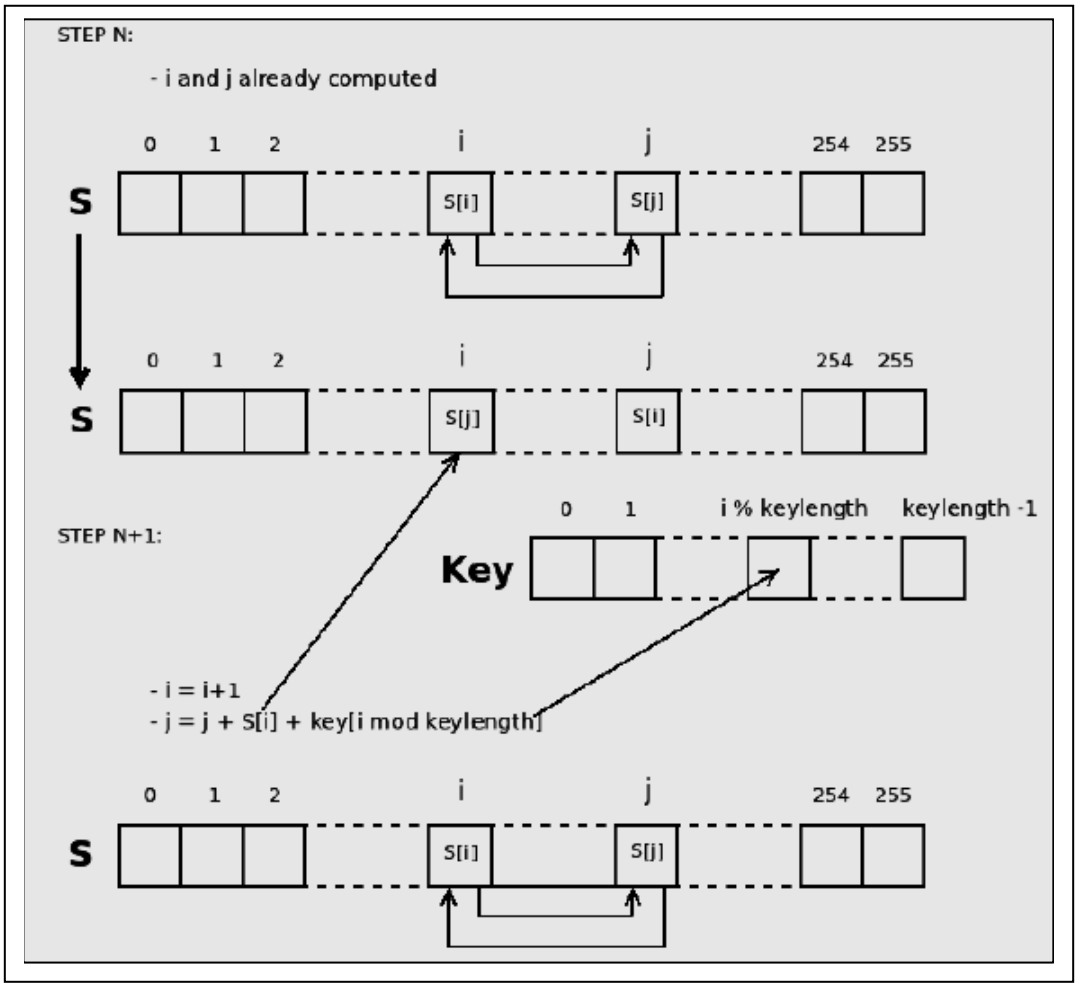

Figure (1) key-Scheduling Algorithm (KSA)

3.2 Pseudo-Random Generation Algorithm (PRGA):

After the "S" array is generated and initialized, the PRGA is the next step of the RC4 algorithm that uses "S" array to generate the keystream where the PRGA works by continually "shuffling" the permutation values of "S" array, and produces bytes of keystream for the encryption process. First step is initialize the two indexes "i" and "j" to "0", then will start the generation of the keystream one byte at a time until reaching the size of the message to encrypt. For

each new byte generating will do the same stepes [12], Algorithm (2), introduces a pseudo-code corresponding to PRGA.

Algorithm (2): Pseudo-code corresponding to PRGA

```
Input: S-array
Output: Stream key (K)
Process:
i , j := 0
While GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
Endwhile
```

Permutation values of "S" array, and produces bytes of keystream for the encryption process are illistrated in Figure (2)

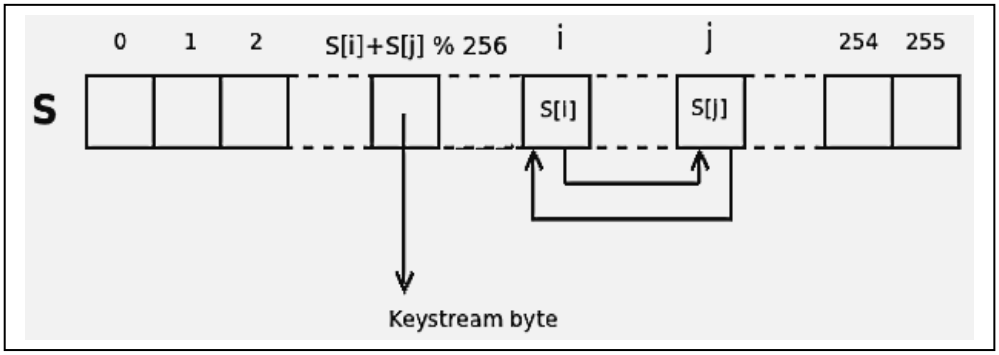

Figure (2) Pseudo-Random Generation Algorithm (PRGA)

Once the keystream has been generated, the encryption process of the plaintext is done simply by XORing the keystream with the plaintext in byte-oriented operations. As for the decryption process, it is as simple as the encryption, and will only have to do the opposite: by XORing the ciphertext with the keystream [12], see Figure (3).

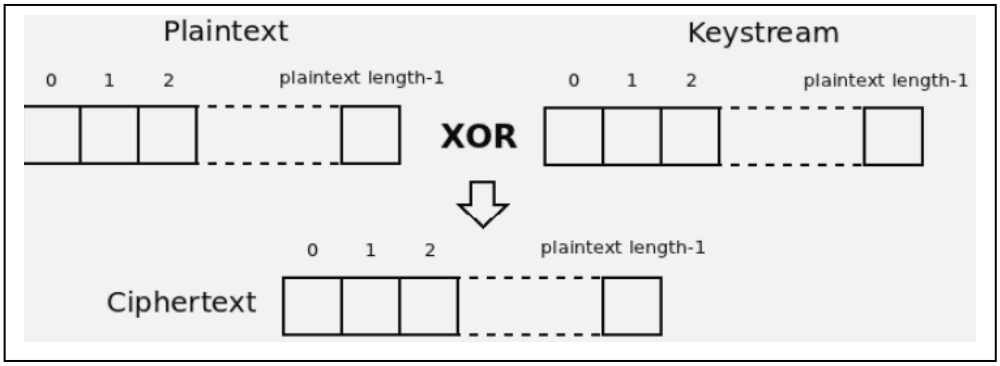

Figure (3-A) RC4 Encryption Process

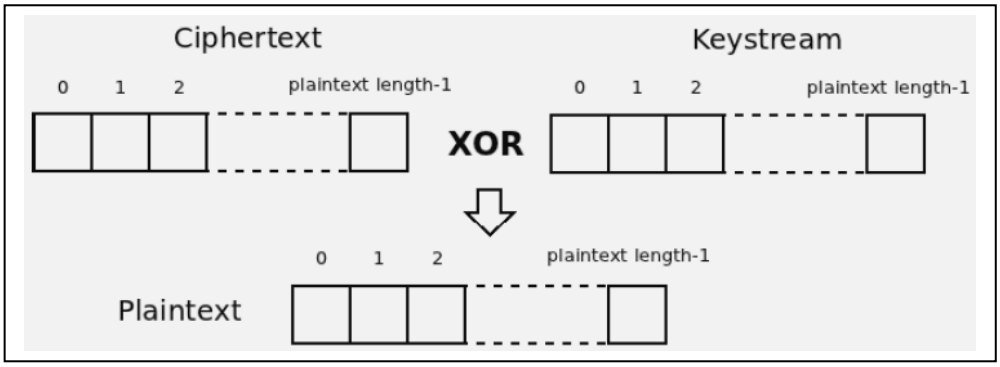

Figure (3-B) RC4 Decryption Process

4. Proposed RC4+S Algorithm

RC4+S is an extension of standard RC4 algorithm. The basic purpose of this enhancement is to making standard RC4 algorithm more strong. RC4 is basically two stages process known as: KSA and PRGA. Weaknesses and attacks are detected in sub processes inside the both stages of RC4 algorithm. The backbone of the RC4 algorithm is shuffling operation in both the stages KSA and PRGA. So making strong RC4 algorithm required to modify the infrastructure of KSA and PRGA, also the proposal introduces a new different sub processes inside the KSA and PRGA.

RC4+S algorithm will try to enhance and improve both stages of RC4 algorithm. Where, RC4+S improves randomness in KSA as well as in PRGA to make them more stronger and overcome

various weaknesses and attacks, a new way is proposed for the KSA and PRGA.

Let's now see the two stages of proposed algorithm, see algorithm (3) and algorithm (4):

Algorithm (3): Proposed KSA (With Double Permutation)

Input: Seed_key (K): [k1, k2, k3, …., k_m]
Output: State_matrix(S)

Initilization of State_matrix(S):

```
For i = 0 to 255
   S [i] = i
End for
```

Permutation of State_matrix(S):

```
j1 = 0
For i = 0 to 255
   j1 = ( j1 + S [i] + K [ i mod keylength ] ) mod 256
   Swap ( S[i] , S[j] )
End for
j2 = 0
For i = 255 to 0
   j2 = ( j2 + S [i] + K [ i mod keylength ] ) mod 256
   Swap ( S[i] , S[j] )
End for
```

Algorithm (4) Proposed PRGA+S (With Dynamic AES S-Box)

Input: State_matrix(S)
Output: Key_stream(K)

Key Stream Processing:

i = 0 ; j = 0

While (plaintext > 0)

i = (i + 1) mod 256

j = (j + S[i]) mod 256

Swap (S[i] , S[j])

T_1 = (S[i] + S[j]) mod 256; T_2 = Dynamic AES-S-box (T_1)

T_3 = (S[i>>3] + S[j]) mod 256

T_4 = (S[i] + S[j<<5]) mod 256

K = [(T_1 XOR T_2) mod 256 + ((T_3 + T_4) Mod 256)] mod 256

plaintext = plaintext -1

End While

4.1 RC4+S Algorithm Description:

Proposed KSA: (With Double Permutation)

The proposed modification design for KSA; which is called (Proposed KSA). KSA trend to cancel the flaws of KSA part in RC4 encryption algorithm by implementing a twice permutation operation for the internal state_matrix(S). The purpose of twice permutaion operation is to minimize and destroy the correlation among state_matrix(S) bytes results from the first permutation operation (P1), which is exploited again as an input to state_matrix(S) for the second permutaintion operation (P2). Therefore the two operations with each other's (P1) and (P2) work on enhance RC4 algorithm and strength state_matrix(S) by

randomize it that is used as input for second stage of algorithm (Proposed PRGA+S).

Proposed PRGA+S: (With Dynamic AES S-Box)

The proposed modification design for PRGA, which is called (Proposed PRGA+S) that trend to cancel the existing correlation problem between the key stream outputs and state_matrix(S) of PRGA. PRGA+S, also maximizes the randomness by applying the diffusion process by using the value of (T_1) as input to generate the value of (T_2) using dynamic Rijndael Substitution table (Dynamic AES-Sbox) then implement the confusion operation by rotating the value of (i) three positions to the right to generate the value of (T_3), and rotate the value of (j) five position to the left to generate the value of (T_4). At last the key stream (K) is resulted from mixing all values of (T_1, T_2, T_3, T_4) to obtain total distribution of key; which is random and non-uniformity.

The proposal, RC4+S algorithm introduces more randomness using the suggested hybrid operation; which is the mixing between "confusion" and the value of random permutation which based on the indicator random position value (i),(j), "diffusion" operation which is not provided by the standard RC4 algorithm. The proposed RC4+S may exhaust more of time and computing power if it compared to standard RC4 algorithm, but from the point of view of security and standard rules of privacy perspective, especially in the applications with the field that deal with private and sensitive data such as payment and e-business transactions, the exhaustive of time and computing power is not a big problem as measured by the significance of data transfer over network.

4.2 Proposed RC4+S Encryption/Decryption Processes

In this algorithm, the encryption and decryption is done in the same manner as standard RC4 algorithm, Figure (4) describes the encryption and decryption processes of the proposed RC4+S algorithm. That shows, the proposed RC4+S algorithm generates

the key stream output bytes and XORing these bytes with the plaintext in character by character manner.

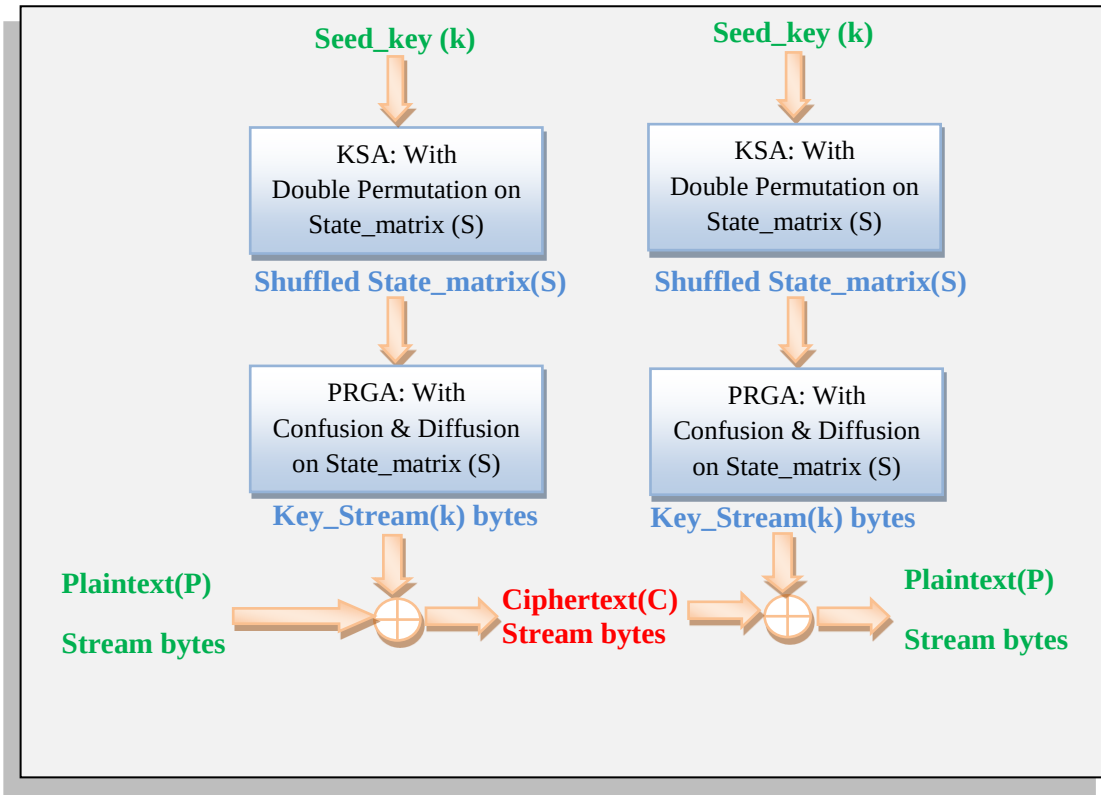

Figure (4) Proposed RC4+S Encryption/Decryption Processes

4.3 Proposed WEP Protocol Encryption Process

Figure (5) describes the encryption process of the proposed WEP protocol using Proposed RC4+S algorithm and CRC-32 message checksum.

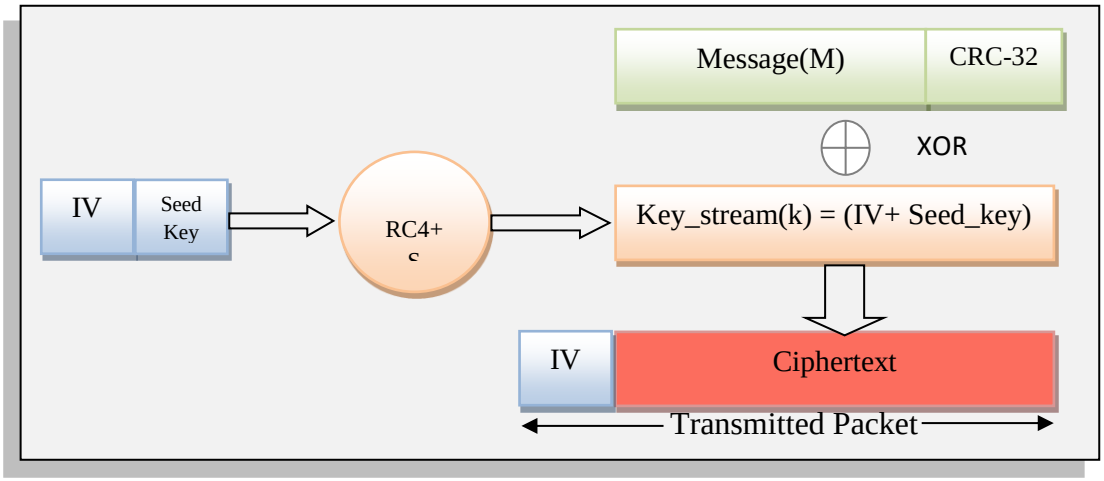

Figure (5) Proposed WEP Protocol Encryption Process

5. Proposed RC4+S Algorithm Evaluation

As performance evaluation, the generated sequence-bits (seed, secret-key, cipher text) of "Proposed RC4+S" will be tested by "basic five statistical tests": (frequency, run, poker, serial, and correlation), that measure the randomness of the output sequences of true random number generators or pseudo-random number generators.

The numeric value of statistical test for secret key and ciphertext of standard RC4 and proposed RC4+S is introduced in:

1. Table (1) (with all its related figures from figure (6) to Figure (10)).
2. Table (2) (with all its related figures from Figure (11) to figure (15)).

- The table results are evaluated by reference to the "Chi-squared distribution" that commonly used to compare observed data (seed, secret-key, cipher text) with data we would expect to obtain according to a specific hypothesis.

- The table results show the "goodness of fit" of data represented by the p-value, the testing process uses "Chi-square distribution" in comparison of the p-value with the significance level (α); in this evaluation (α = 0.05).
- The average data size is between 20-30 bytes (160- 240 bits).
- The row (AVG): shows the average of data for each test type of table columns.
- The row (DIFF): shows the percentage of difference between standard RC4 and proposed RC4+S algorithm for each test type of table columns.
- The rate of difference (improvement) for Secret-keys randomness in proposed RC4+S algorithm is estimated by approximately (20%) at an average rate.
- The rate of difference (improvement) for Ciphertext randomness in proposed RC4+S algorithm is estimated by approximately (15%) at an average rate.
- Although the percentage rate of difference (improvement) for the Secret-keys of the proposed RC4+s algorithm is (20%), But it goes down to approximately (15%) for the Ciphertext, due to using fixed plaintext which negatively affects the ratio.