International Journal of Computing and Network Technology
ISSN 2210-1519
Int. J. Com. Net. Teach. 3, No. 2 (May 2015)

Enhancing Security and Speed of RC4

Maytham M. Hammood[1, 2], Kenji Yoshigoe[1]and Ali M. Sagheer[3]

[1] Department of Computer Science, University of Arkansas at Little Rock, Little Rock, USA
[2]Department of Computer Science and Mathematics, University of Tikrit, Iraq
[3]Computer College, University of Anbar, Iraq

Received: 10 Dec. 2014, Revised: 15 March, 2015, Accepted: 1 April 2015, Published: 1 (May) 2015

Abstract: Wireless communication security is a critical factor for secure communication among large scale of wireless networks. A limited resource constraint, such as power and memory size presents a significant challenge to implement existing cryptographic algorithms. One of the most important symmetric cryptographic algorithms is Rivest Cipher 4 (RC4) stream cipherthatis utilized in many real-time security applications. However, the RC4 cipher shows some weaknesses, including a correlation problem between the public known outputs of the internal state. In this paper, we propose RC4 stream cipher with a random initial state (RRC4) to solve the weak keys problem of the RC4 using a random initialization of internal state *S*. We also propose RC4 stream cipher with two state tables (RC4-2S) to solve the correlation problem between the public known outputs of the internal state using permutation between state1 (S_1) and state 2 (S_2) while requiring less time than RC4. Finally, we propose RC4 stream cipher with two state tables togenerate four keys (RC4-2S$^+$) in each cycle which further enhances randomness overRC4-2S and RRC4.

Keywords: Encryption, Stream Cipher, RC4, Random Number Generator

1. INTRODUCTION

Cryptographic algorithms that can provide fast implementation, small size, low complexity, and high security for resource-constrained devices such as wireless sensor devices are imperative. Conventional cryptographic algorithms are sequences of processes or rules, used to encrypt or decrypt messages in a cryptographic system to provide security services. Conventional cryptographic algorithms are very complex and consume a significant amount of power when used by resource-constrained devices for the provision of secure communication [1]. These include symmetric and asymmetric key algorithms, but asymmetric key algorithms are inappropriate for resource-constrained devices for several reasons, including the limited storage space and excessive energy usage [1, 2]. Therefore, security schemes should rely on symmetric key cryptography, especially when systems have limited hardware resources.

Stream ciphers can be classified as synchronous and asynchronous. In a synchronous stream cipher, a sequence of keys is generated independently from the plaintext and ciphertext. The drawback of synchronous is that both the sender and the receiver have to be synchronized for key usages. It can detect any deletion or insertion of bits by an active attack directly, yet such an attack can cause immediate loss of synchronization [3]. Asynchronous stream ciphers depend on the previously generated ciphertext in order to continue generating new ones, yet it cannot regenerate the same sequence of ciphertext. On the other hand, synchronous stream ciphers can regenerate a sequence of ciphertext because a key sequence is independent of the plaintext. This case is referred to as self-synchronized because of the ability to re-synchronize after insertion or deletion of bits. Although self-synchronization can be an important property, this type of the stream cipher has not received much attention [4].

In the literature, there are many stream cipher algorithms presented to implement high-performance software. Such algorithms include the following: A5/1 [5], ORYX [35], LEVIATHAN [36], IDEA [37], MUGI [38], W7 [40], RC4, Helix [39], SNOW [32], SOBER [34], and SEAL [33]. One-time pad, which is stronger than all of these aforementioned algorithms, is unbreakable since it never uses the same key more than once. Consequently, the sequence key is truly random and not generated in a deterministic manner as is the case with other algorithms. Thus, an attacker cannot deduce any information about the key from ciphertext [6]. The robustness of stream ciphers depends on Pseudo-Random Number Generator (PRNG) which has proved to be resistant to attacks if it passes the statistical tests.

E-mail: mmhammood@ualr.edu, kxyoshigoe@ualr.edu, ali.sagheer@gmail.com

RC4 is a proprietary stream cipher that was designed in 1987 by Ron Rivest. RC4 is widely used in security software based on the stream cipher, including one in the encryption of traffic to and from secure websites such as Transport Layer Security (TLS), Secure Socket Layer (SSL), and Wired Equivalent Privacy (WEP) implementations. RC4 is fast in comparison to other algorithms, and it has a simple design hardware implementation [7]. In addition, RC4 is five times faster than Data Encryption Standard (DES) and fifteen times faster than Triple-DES [4].

Many stream ciphers use a de-facto RC4 standard for meeting specific requirements, such as the limited storage size and power of the devices; however, there are many weaknesses found in stream ciphers caused by mathematical relationships between the key, ciphertext, and plaintext which attackers can use to assess the security of the cryptographic algorithms via cryptanalysis. Thus, the goal is to create a sequence of keys that approaches true randomness [8].

The structures of stream ciphers are presented more than block ciphers. The security of the cryptographic algorithms can be assessed by cryptanalysis. Many weaknesses found in stream ciphers are triggered by mathematical relations between the key, ciphertext, and plaintext. The primary aim is to produce a random key asymptotic to the true random key [8]. Cryptanalysis refers to theoretical knowledge and uses numerous statistical tools to find weaknesses in the ciphertext or cryptosystems.

The rest of the paper is organized as follows. Section 2 reviewsrelated works. Section 3 presents the description of RC4, and Section 4 shows some weaknesses of RC4. We then present our proposed algorithms to enhance RC4 algorithm in section 5. Section 6 describes the evaluation, and Section 7 presents the results of the analysis. Section 8 concludes the paper.

2. RELATED WORK

Many researchers have tried to enhance the security of RC4 and create variant algorithms. However, this enhancement slowed the implementation speed. On the other hand, many researchers have attempted to improve algorithmic speed, but this caused a decrease in the randomness. Zoltak [8] proposed Variably Modified Permutation Composition (VMPC) which was designed to be efficient in software implementations and solved a weakness found in the RC4 Key Scheduling Algorithm (KSA) that had been described by Fluhrer et al.[9].The structure of Pseudo-Random Generation Algorithm (PRGA) in VMPC was more complex in comparison with the RC4 which makes it more resistant to attacks. Paul and Preneel [10] presented RC4A as an enhancement over RC4 after exploring a new statistical weakness in the first two output bytes of the RC4 key stream generator. They argued that the number of outputs required for distinguishing the output of the RC4 random sequence with the presence of bias is 128, and they recommended the use of 256 to overcome this bias. RC4A is considered to be robust against most of the weaknesses of RC4, particularly its weakness of distribution in the first two output bytes. However, after one year, Maximov [11] proposed distinguishing attack on both VMPC and RC4A, which can distinguish the cipher output from a truly random number.

Yu and Zhang [12] presented RC4 state combined with the hash function without affecting the simplicity and efficiency. The RC4 state based on the hash function can generate Message Authentication Code (MAC). The enhancement includes the offset, forward, and backward properties of RC4 states where the authors use offset to ignore the first few bytes of the key and started encrypting the data in determining the position which has led to an increase in the time of execution. Pardeep and Pateriya [13] proposed PardeepCipher-RC4 (PC-RC4) as an extension to the RC4 algorithm to improve randomness in KSA and PRGA, yet it increased the execution time. Kamble and Meshram found that RC4 had a weakness to differential attack and showed how the plain text can be retrieved. In addition, the authors recommended how to avoid this attack using different sizes of key of greater than 32 bits [14].

Kadry and Smaili [15] presented Vigenère RC4 (VRC4) which is a combination of the RC4 and the poly alphabetic cipher Vigenère. The plaintext is encrypted using the classic RC4 cipher followed by re-encrypting by Vigenère, which results in increased time of execution. Mironov [16] presented a new model of RC4 and analyzed it by applying the theory of the random permutation. Based on this study, he recommended discarding at least the first 512 bytes to avoid those weaknesses which have led to an increase in the execution time.Mousa and Hamad examined the analysis of the effect of different parameters of the RC4 algorithm such as the execution time and the file size and concluded that the speed of encryption and decryption time is affected by the length of the encryption key and the size of the data file [17].Yao et al. presented analysis and enhancement of the security of the RC4 algorithm by using public key encryption with RC4, which has led to an increase in the size of the system and the time of execution [18].

Chefranov and Mazurova [19] presented an extension of RC4 (RC4E) in which they combined RC4 with a simple Heap's algorithm to provide enumeration of all potential permutations for periodic change of RC4 state. However, their algorithm requires additional memory to keep one more permutation compared with RC4. RC4$^+$

presented by S. Maitra and G. Paul [31] consists of three layers to avoid correlations and increase the randomness of RC4. It has a more complex output function which performs four additional lookups for each byte output. The running time of KSA^+ is around three times slower than that of RC4 KSA, and $PRGA^+$ is approximately 1.70 times slower than RC4 PRGA.

Hammood et al. proposed an RRC4 random initial state algorithm with a new enhancement of RC4, and improved randomness of the traditional RC4 [20],andproposed an RC4 with two state tables (RC4-2S) which has improved the key generation time of the conventional RC4 while also outperforming randomness of the keys generated [21].

3. DESCRIPTION OF RC4

RC4 design avoids the use of Linear Feedback Shift Registers (LFSRs), which many stream cipher algorithms depend on, especially in hardware. The idea of RC4 is to manipulate the elements by swapping them to achieve superior randomness. The RC4 algorithm has a variable key length, which ranges between 0 to 255 bytes for initializing 256-byte array in the initial state by elements from *S* [0] to *S* [255]. As recommended in [9, 22] RC4 must use a key longer than 128 bytes. The algorithm consists of KSA and PRGA, which are executed sequentially. RC4 Key is initialized by KSA while the pseudo-random number is produced by PRGA. The pseudo codes for the two parts of the RC4 algorithm are shown in Algorithm 1 and Algorithm 2 wherem is the message length of plaintext,*L* is the length of the initial key in bytes,*N* is the size of the array or state*S*, and*i* and*j* are indexed pointers. Such a self-modifying lookup table is simple and has been applied efficiently in software [23]. The output of the PRGA algorithm is a key sequence that will be XOR-ed with plaintext (or ciphertext) to get the ciphertext (or plaintext).

Algorithm 1. Key Scheduling Algorithm (KSA) for RC4

INPUT: $K[k_1, k_2, \ldots, k_n], m$
OUTPUT: State $S=S_0, S_1, \ldots, S_{255}$
1. $S[i]=i$, for $i=0, 1, 2, \ldots, 255$
2. $j \leftarrow 0$
3. For $i \leftarrow 0$ to 255 Do
   3.1. $j \leftarrow (j+S[i]+K[i \bmod keylength]) \bmod 256$
   3.2. Swap S[i] withS[j]
4. Return(S)

Algorithm 2. Pseudo-Random Generation Algorithm(PRGA) for RC4

INPUT: State S
OUTPUT: Key sequence Kseq
1. $j \leftarrow 0$
2. $i \leftarrow 0$
3. While not end of sequence Do
   3.1. $i \leftarrow (i+1) \bmod 256$
   3.2. $j \leftarrow (j+S[i]) \bmod 256$
   3.3. SwapS[i] withS[j]
   3.4. $Kseq \leftarrow S[(S[i]+S[j]) \bmod 256]$
4. Return(Kseq)

S. O. Sharif and S.P. Mansoor provided a comparison between different encryption algorithms using different data sizes and key sizes. The simulation results clarify preponderance of the RC4 algorithm over other algorithms in terms of speed and throughput [24] as shown in Fig. 1 and 2.

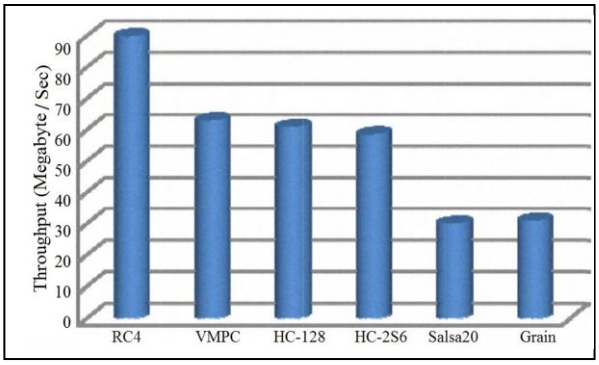

Figure 1. Encryption and Decryption time for stream cipher [24]

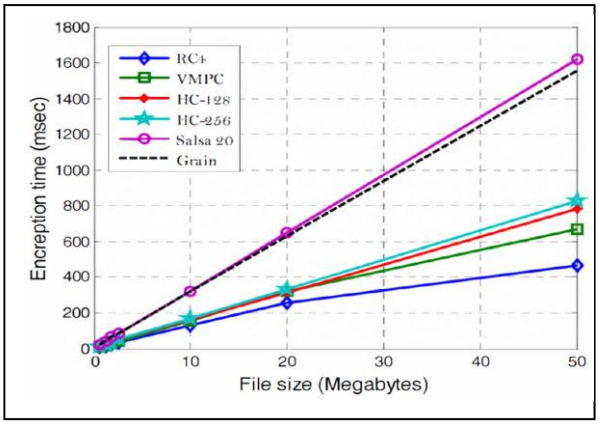

Figure 2. Throughput of the Stream Cipher [24]

N. Singhal and J.P.S.Raina [30] conducted a performance evaluation of the AES algorithm and RC4 algorithm.The performance metrics were the time of the CPU, encryption\decryption time, memory utilization, throughput and size of the key. The results exhibited that the RC4 is faster than AES and more energy efficient for encryption and decryption. In addition, the AES consumes more memory.

4. THE WEAKNESSES OF RC4

After RC4 had been revealed to the public in 1994, interest in its cryptanalysis grew swiftly, especially after it was used for the Wired Equivalent Privacy (WEP) implementations. As a result, several weaknesses in RC4 were identified, including the one in KSA and one in the relations between the S-box in different time. Some of these weaknesses are simple and can be solvedeasily, but others are critical because they can be exploited by attackers. Initial state of the PRGA achieves a pretty good efficiency against a number of attempted attacks.Mantin and Shamir [25] distinguished statistical weakness in the RC4 algorithm that the probability of the second round output byte generated is zero output bytes is twice than other values, which can exploit this weakness by the practical ciphertext-only attack. This ciphertext-only attack is limited to broadcast applications in which different keys are used to encrypt the same plaintext for multiple recipients. Since a fairly straightforward approach such as a brute force attack infers the internal state of the PRGA, increased internal state size is recommended, yet it results in increased encryption and decryption time.

Roos [26] studied the weaknesses of RC4 and found a significant correlation between the first few values in the state table and generated value. The main reason for that is the state table is first initialized with the sequence (0, 1, 2,..., 255). For at least one out of every 256 possible keys, the initial byte of the key generated by RC4 is strongly correlated with a few bytes of the key. Occasionally, therefore, the keys allow prediction of the first bytesof the output of the PRGA. To eliminate this problem, has been recommended to reject at least the first 256 bytes of the key stream output to discourage this attack [16].

Our algorithms will address this problem without the need to neglect the first byte of output by increasing the permutation between the elements or use a random initial state. This weakness is caused by the utilization of the regular sequence of numbers from0 to255 as the initial state. This gives the attacker a starting point to track numbers because the initial state has a regular sequence. Furthermore, there is a significant statistical bias in the first output bytes which can be used to mount a ciphertext-only attack [9]. There are many other attacks described in [9] such as a sub-key guessing attack, linear consistency attack, inversion attack, etc. In addition, an algebraic attack is a new type of higher order correlation attack

5. NEW METHODS

RC4 has a significant number of weaknesses in the phases of KSA and PRGA. Many researchers have tried to solve these problems by creating variant algorithms. However, there is still an issue in the randomness b or the speed of these algorithms. We have attempted to increase the randomness as well as increase the speed of implementation. In our methods, all addition operations are carried out modulo N or $N/2$. KSA takes a key k consisting of one n-bit word. After the setup, the round algorithm is executed once for each word output. In reality, all practical applications of the developed RC4 are implemented with$n = 8$ in which case all the entries of S along with$i$ and$j$ are bytes:

A. RC4 Stream Cipher with A Random Initial State (RRC4)

RC4 has a significant number of weaknesses initsKSA and PRGA phases. One of the causes of these weaknesses is that the initialization process produces a deterministic sequence (0, 1, 2, ... 255) which gives anadvantage to the attacker from the first place.We describe theRC4 stream cipher with a random initial state (RRC4), recently proposed in [20],to improve randomness of keystream generated by RC4. Since the correlation among the publicly known outputs of the internal state depends on the randomness of the key sequence, RRC4 further tries to reduce the correlation problem that various RC4 attacks target by exploiting strong correlation among the publicly known outputs of the internal state.

RRC4 simplyconsists ofa revised KSA (as shown in Algorithm 3) followed by the PRGA of RC4.

In KSA phase, the content of Sconsisting of N values of 0 to $N-1$ without duplicationcan be saturated via aid of any pseudo-random number generator for later distribution to the receiver as an additional secret key. The system can re-generate new random initial state table and replace it with the previous one periodically as needed.

The input secret key k is used as a seed for the state$S$. In the systems or implementations which dynamically update$k$ between a sender and a receiver,S and$k$ can be shared at the same time.Therefore,S becomes a secret random input to the PRGA phase as shown in Algorithm 2. S in PRGA is used to produce a sequence of the output stream which will be XOR-ed with plaintext to get ciphertext or XOR-ed with ciphertext to get plaintext.

Algorithm 3. Key Scheduling for RRC4

INPUT: Keys k,n
OUTPUT: S
1. *For i ← 0 to N - 1 Do*
   S[i] ← Random$_i$
   where S∩S=S∪S=S={0, 1, 2, 3, 4… N-1}
2. *j← 0*
3. *For i ← 0 to N - 1 Do {*
   3.1. j ← (j + S[i] + k [i mod keylength]) mod N
   3.2. Swap S[i] with S[j] }
4. *j← 0*
5. *Return (S)*

B. RC4 Stream Cipher with Two State Tables (RC4-2S)

The RC4 stream cipher with two state tables (RC4-2S), recently proposed in [21],was designed to reduce the correlation problem between the public known outputs of the internal state while also improving the speed of encryption and decryption.RC4-2Sconsists of KSA and PRGA phases as shown in Algorithms 4 and 5, respectively.

In KSA phase, S_1 is filled with values ranging from0 to (*N/2*)-1 while$S_2$is filled with the remaining *N/2* numbers from *N/2* to *N-1*. The input secret key, *k*, as a seed, is used to make permutations and swapping of the elements of S_1 and S_2. Consequently, S_1 and S_2 become two secret random inputs for PRGA phase.

In PRGA phase, S_1 and S_2 produce two keys in each key generationcycle (instead of one byRC4). In this algorithm, there are more elements to be swapped between S_1 and S_2 by three pointers: i, $j_1 = j_1 + S_1[i]$, and $j_2 = j_2 + S_2[i]$ in the *S-box*. S_1 and S_2 in PRGA are used to produce the sequence of output stream , which is XOR-ed with data to encrypt data or XOR-ed with the ciphertext to decrypt data.

Algorithm 4. Key Scheduling forRC4-2S

INPUT: Keys k, n
OUTPUT: S_1, S_2
1. *For i ← 0 to N/2 - 1 Do*
   $S_1[i] \leftarrow i$
2. *For i ← N/2 to N - 1 Do*
   $S_2[i-N/2] \leftarrow i$
3. *j← 0*
4. *For i ← 0 to N/2 - 1 Do {*
   4.1. $j \leftarrow (j + S_1[(i+ k [i \bmod L]) \bmod N/2] + k [i \bmod L]) \bmod N/2$
   4.2. Swap $S_1[i]$ with $S_1[j]$ }
   4.3. j← 0
5. *For i ← 0 to N/2 - 1 Do {*
   5.1. $j \leftarrow (j + S_2[i] + k [i \bmod L]) \bmod N/2$
   5.2. Swap $S_2[i]$ with $S_2[j]$}
6. *Return (S_1 and S_2)*

Algorithm 5. PRGAforRC4-2S

INPUT: S_1, S_2.
OUTPUT: Key sequence Kseq
1. *$i, j_1, j_2 \leftarrow 0$.*
2. *While not end of half sequence Do*
   2.1 $i \leftarrow (i+1) \bmod N/2$
   2.2 $j_1 \leftarrow (j_1+S_1[i]) \bmod N/2$
   2.3 Swap $S_1[i]$ with $S_2[j_1]$
   2.4 $t_1 \leftarrow S_1[(S_1[i] +S_1[j_1]) \bmod N/2]$
   2.5 $j_2 \leftarrow j_2 + S_2[i] \bmod N/2$
   2.6 Swap ($S_2[i]$ with $S_1[j_2]$)
   2.7 $t_2 \leftarrow S_2[(S_2[i] + S_2[j_2]) \bmod N/2]$
   2.8 Swap ($S_1[t_1]$ with $S_2[t_2]$)
   2.9 Kseq ← $[t_1, t_2]$
3. *Return (Kseq)*

C. RC4 Stream Cipher with Two State Tables to generate 4 Keys (RC4-2S$^+$)

Many researchers have attempted to enhance the security of RC4, which have resulted in added complexity and delay. Others have tried to improve the speed of the cipher, which have weaken the randomness of the generated keys. RC4-2S$^+$ is designed to improve the randomness of the generated keys without impacting the speed of RC4 by generating four keys in each of the key generation cycle The RC4-2S$^+$ algorithm consists of KSA and an output phase (PRGA) as shown in Algorithms 4 and 6, respectively.

In KSA, S_1 is filled with values ranging from 0 to (*N/2*)-1 while$S_2$is filled with the remaining *N/2* numbers from *N/2* to *N-1*. The input secret key, *k*, as a seed, is used to make permutations and swapping of the elements of S_1 and S_2. Consequently, S_1 and S_2 become two secret random inputs for PRGA phase.

In PRGA, S_1 and S_2 produce four keys in each of the key generation cycle instead of one or two keys by RC4 and RC4-2S, respectively. In this algorithm, there are four swapping operations between S_1 and S_2 by five pointers: i, $j_1 = j_1 + S_1[i]$, $j_2 = j_2 + S_2[i]$, $t_2 = S_2[(S_2[i]+ S_2[j_2])]$, and $t_3 = S_1[(S_2[i]+ S_1[j_2])]$, in each cycle of the *S-box*. S_1 and S_2 in PRGA are used to produce the sequence of theoutput stream which is XOR-ed with plaintext (or ciphertext) to generate the ciphertext (or plaintext).

Algorithm 6. PRGA forRC4-2S$^+$

INPUT: S_1, S_2.
OUTPUT: Key sequence Kseq
1. *$i, j_1, j_2 \leftarrow 0$*
2. *While not end of quarter sequence Do*
   2.1 $i \leftarrow (i+1) \bmod N/2$
   2.2 $j_1 \leftarrow (j_1+S_1[i]) \bmod N/2$

2.3 Swap $S_1[i]$ with $S_2[j_1]$
2.4 $t_1 \leftarrow S_1[(S_1[i] + S_1[j_1]) \bmod N/2]$
2.5 $j_2 \leftarrow j_2 + S_2[i] \bmod N/2$
2.6 Swap $S_2[i]$ with $S_1[j_2]$
2.7 $t_2 \leftarrow S_2[(S_2[i] + S_2[j_2]) \bmod N/2]$
2.8 $t_{11} \leftarrow t_1 \bmod N/2$
2.9 $t_{22} \leftarrow t_2 \bmod N/2$
2.10 Swap $S_1[t_{11}]$ with $S_2[t_{22}]$
2.11 $t_3 \leftarrow S_1[(S_2[i] + S_1[j_2]) \bmod N/2]$
2.12 $t_{33} \leftarrow (t_1 + t_2 + t_3) \bmod N/2$
2.13 $t_{44} \leftarrow t_3 \bmod N/2$
2.14 Swap $S_1[t_{33}]$ with $S_2[t_{44}]$
2.15 $t_4 \leftarrow S_2[(S_1[i] + S_2[j_1]) \bmod N/2]$
2.16 $Kseq \leftarrow [t_1, t_3, t_2, t_4]$
3. Return (Kseq)

Therefore, we encrypt four bytes of plaintext each time as:

$$C_i = m_i \oplus t_1 \quad (1)$$

$$C_{i+1} = m_{i+1} \oplus t_3 \quad (2)$$

$$C_{i+2} = m_{i+2} \oplus t_2 \quad (3)$$

$$C_{i+3} = m_{i+3} \oplus t_4 \quad (4)$$

6. EVALUATION

The binary generated sequences were tested by the National Institute of Standards and Technology (NIST) Test Suite. The statistical package of random number generation testing consists of 16 statistical tests to measure the randomness of the output sequences of true random number generators or pseudo-random number generators [27]. The NIST context depends on hypothesis testing. Hypothesis testing is a procedure to determine the probability that a given hypothesis is reasonable by using statistics. NIST test includes tests to determine whether or not a generated sequence of zeroes and ones are random. These tests can detect a diversity of various non-randomness that may well exist in a generation sequence:

A. Randomness Test

Generally, these sixteen tests are categorized into two groups. The first group is called *non-parameterized test* and includes: Frequency Test, Cumulative Sums Test (forward and backward), 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 group, referred to as a *parameterized test*, includes Serial Test, Linear Complexity Test, Overlapping Template of All Ones Test, Non-overlapping Template Matching Test, Approximate Entropy Test, Block Frequency Test, and Universal Statistical Test. A brief description of each test follows:

- Frequency Test: Determines if the ratio of the number of ones and zeros in a generated sequence is approximately the same. It is a worth mention recommendationby NIST that the Frequency test be applied first, since this test provides the most fundamental evidence for whether or not the presence of non-randomness in the generated sequences such as non-uniformity.
- Approximate Entropy Test: Determines whether a sequence appears more regular by comparison with anticipated results from a truly random sequence.
- Block Frequency Test: Determines whether the number of zeros and ones in each m-bit block are approximately the same to have a random distribution.
- Cumulative Sums (Forward) Test: Determines whether or not the maximum of the cumulative sums in a sequence possesses significant zeros or large ones at the beginning of the sequences. This test could be considered a random walk. The results of the random walk must be close to zero.
- Cumulative Sums (Reverse) Test: Determines whether the maximum of the cumulative sums in a sequence contains too many zeros or ones at the end of the sequences. This test could be considered a random walk. The results of the random walk must be close to zero.
- Discrete Fourier Transform Test: Determines the spectral frequency of the binary sequence that would be expected for a truly random sequence.
- Lempel - Ziv Compression Test: Determines whether or not a sequence is more compressed than a truly random sequence.
- Linear Complexity Test: Assesses whether the generated sequence is complex enough to be considered random or not.
- Longest Run of One's Test: Assesses the distribution of long runs of ones within an m-bitblock to determine if it agrees with the theoretical probabilities.
- Non-periodic Templates Test: Determines whether or not there are too many occurrences of non-periodic templates.

- Overlapping Templates of All One's Test: Determines the frequency of *m*-bit periodic templates.
- Random Excursions Test: Assesses whether the distribution of the quantity of calls of a random walk to a particular state.
- Random Excursions Variant Test: Assesses the distribution of the entire amount of visits across many random walks to a particular state and concludes whether or not it has exceeded the truly random sequence.
- Rank Test: Assesses rank distribution of the corresponding random sequence due to the periodicity of sub-sequences that repeat.
- Runs Test: Assesses whether the entire number of runs denotes that the frequency in the bit stream is very fast or very slow.
- Serial Test: Assesses whether the number of occurrences of m-bit overlapping patterns is approximately the same.
- Universal Statistical Test: Assesses the compressibility by determining whether or not the binary sequence can be considerably compressed without loss of information.

B. *Configuration*

The design of the proposed was done using Visual C^{++} and the tests of this PRNG were conducted using NIST STS-1.6. The probability of good or bad random number generator is represented by *p*-value. The testing process compared *p*-value to 0.01. If the *p*-value is more than 0.01, then the process accepts the sequence; otherwise, it rejects the sequence because the sequence is not random. Conversely, some tests accept large sizes of sequence and others fail in a small size; still other tests accept both sizes.

If the tests give a *p*-value asymptotically to 1, then the sequence appears to have perfect randomness. A *p*-value of zero indicates that the sequence appears to be completely non-random. The *SUCCESS* indicates the sequence is acceptable and has enough randomness, whereas the *FAILURE* indicates that the sequence is not acceptable due to non-randomness.

In the NIST statistical suite, there are two tests (Random Excursion and Random Excursion Variant) that do not provide results each running because these tests give a result when the number of cycles exceeds 500. Where, the random hypothesis is depending on the number of cycles in the sequence as shown in equation 5 if the equation is truethat is mean doesn't give the result [29].

$$J < max(0.005\sqrt{n}, 500) \quad (5)$$

Where *J* is the total number of cycles in the sequence and when the *p*-value is small, the random hypothesis is rejected. Therefore, the test results are shown as a text file in NIST. The results of the random excursion test and the random excursion variant test are shown in Fig. 3 and 4 respectively.

```
            RANDOM EXCURSIONS TEST
 ---------------------------------------------
 COMPUTATIONAL INFORMATION:
 ---------------------------------------------
 (a) Number Of Cycles (J) = 0150
 (b) Sequence Length (n)  = 1072000
 ---------------------------------------------
 WARNING:  TEST NOT APPLICABLE.  THERE ARE AN
           INSUFFICIENT NUMBER OF CYCLES.
 ---------------------------------------------
```

Figure 3. Screenshot of Stats.txt file in NIST for random excursion test

```
         RANDOM EXCURSIONS VARIANT TEST
 ---------------------------------------------
 COMPUTATIONAL INFORMATION:
 ---------------------------------------------
 (a) Number Of Cycles (J) = 150
 (b) Sequence Length (n)  = 1072000
 ---------------------------------------------

 WARNING:  TEST NOT APPLICABLE.  THERE ARE AN
           INSUFFICIENT NUMBER OF CYCLES.
 ---------------------------------------------
```

Figure 4. Screenshot of Stats.txt file in NIST for random excursion variant test

7. RESULTS

In our program, we used a large size 134,000 bytes (1,072,000 bits) generated by each key. These sequences were tested, and we subsequently calculated the average of the *p*-values resulting from these tests. As shown in Table 1, the *p*-values are acceptable when greater than 0.01, and the produced sequence can be deemed random, uniformly distributed and suitable for cryptography.

A. *RRC4*

This technique solves the weak keys problem by using initial randomization of *S*. As shown in Fig. 5 the output of RRC4 is more randomness than RC4.

B. *RC4-2S*

This technique solves the correlation between the public known outputs of the internal state.RC4-2S is better than RC4 in most of the NIST tests as shown in Fig.6. Moreover, it's faster than RC4 because RC4-2S requires two swaps and five modulo functions to generate two bytes of key per iteration in the PRGA algorithm while RC4 requires one swap and three modulo functions to generate only one byte of the key.

TABLE I. THE RESULTS OF RUNNING THE NIST SITE OVER THE SET DATA PRODUCED BY THE PROPOSED RC4-2S, RRC4, RC4-2S$^+$ AND RC4A WITH STANDARD RC4

Test No.	Name of Statistical Test	RC4	RC4-2S	RRC4	RC4-2S$^+$	RC4A
		P-VALUE	*P-VALUE*	*P-VALUE*	*P-VALUE*	*P-VALUE*
1	Approximate Entropy	0.49167	0.22135	0.50085	0.45257	0.44579
2	Block Frequency	0.48946	0.59337	0.51171	0.60949	0.52911
3	Cumulative Sums (Forward)	0.51176	0.55508	0.55566	0.56419	0.46738
4	Cumulative Sum (Reverse)	0.50993	0.53429	0.56214	0.61422	0.4937
5	FFT	0.49877	0.51413	0.47113	0.55594	0.5196
6	Frequency	0.50632	0.53623	0.55633	0.55185	0.4456
7	Lempel-Ziv compression	1	1	1	1	1
8	Linear Complexity	0.50012	0.50293	0.47445	0.53915	0.55512
9	Longest Runs	0.46033	0.47262	0.49413	0.4785	0.43785
10	Nonperiodic Templates	0.5005	0.4997	0.50374	0.49981	0.50006
11	Overlapping Template	0.49631	0.47713	0.43561	0.46976	0.50154
12	Random Excursions	0.46179	0.49832	0.50801	0.50867	0.49938
13	Random Excursions Variant	0.49698	0.64245	0.52241	0.50545	0.51135
14	Rank	0.45641	0.47568	0.53163	0.46103	0.49098
15	Runs	0.45766	0.49993	0.57525	0.51482	0.5276
16	Serial	0.53627	0.42557	0.55914	0.48328	0.50561
17	Universal Statistical	0.44057	0.54208	0.48426	0.52704	0.4397

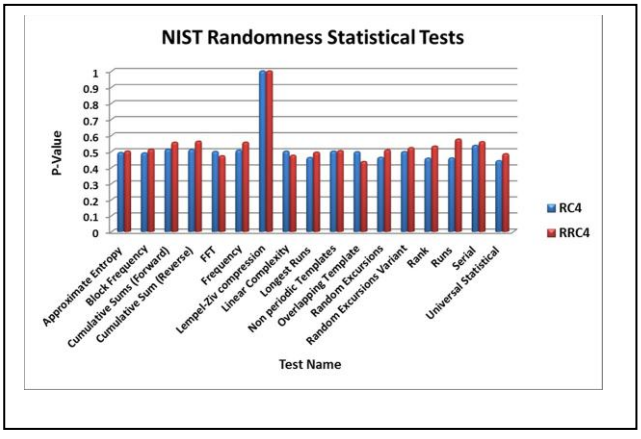

Figure 5. NIST Statistical Test comparison between RC4 and RRC4

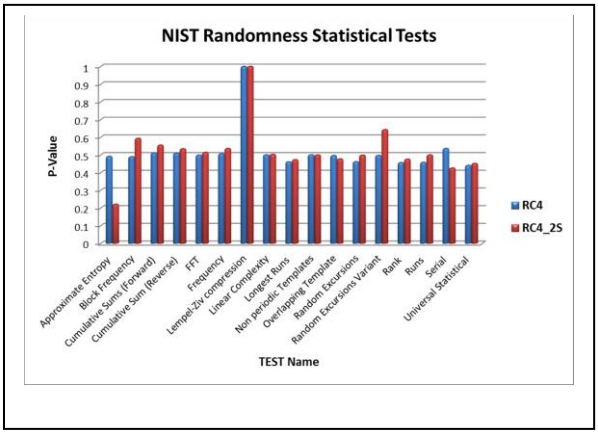

Figure 6. NIST Statistical Test comparison between RC4 and RC4-2S

RC4-2S^{+}

This technique increases the randomness of the initial internal states with permutation of the two state tables during key generation to solve the correlation problem between the public known outputs of the internal. The output sequence, which generated by RC4-2S^{+} is, possessed high randomness compared with the RC4 as shown in Fig. 7.

C. Overall Results

There are some statistical tests of PRNG that are very common and must be included in test suites, such as the Runs test, Frequency test, and Universal test [28]. In these tests, the *p*-values of the proposed algorithms are greater than the *p*-values of the standard RC4, as shown in Fig. 8. We have also tested RC4A algorithm, which is known to be resistant against most known attacks on RC4 [10]. Our test results showed the proposed algorithms have improved randomness over RC4A as shown in Fig.9.

It is worth mention that RRC4 and RC4-2S^{+} performed better randomness than standard RC4 and RC4-2S. However, the key generation time of RC4-2S was faster than that of the original RC4 and the others by approximately 20% for different amounts of keys generated as shown in Table 2 and Fig. 10.

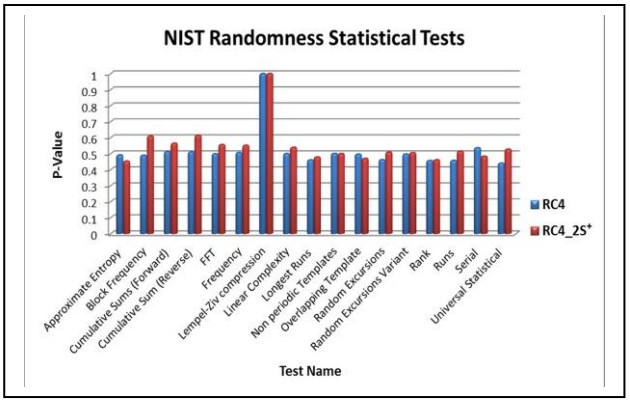

Figure 7. NIST Statistical Test comparison between RC4 and RC4-2S^{+}

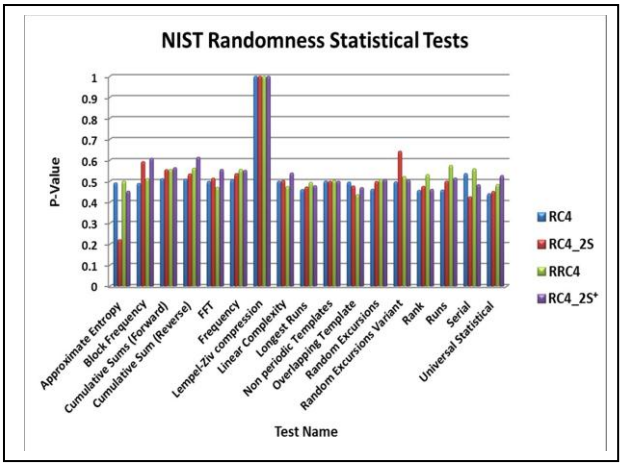

Figure 8. NIST Statistical Testfor all algorithms

TABLE II. KEY GENERATION TIME FOR RC4, RC4-2S

Amount of Key in KB	*RC4 (ms)*	*RC4-2S*
100	232.726	189.4
500	1372.36	970.195
1000	2407.425	1932.196

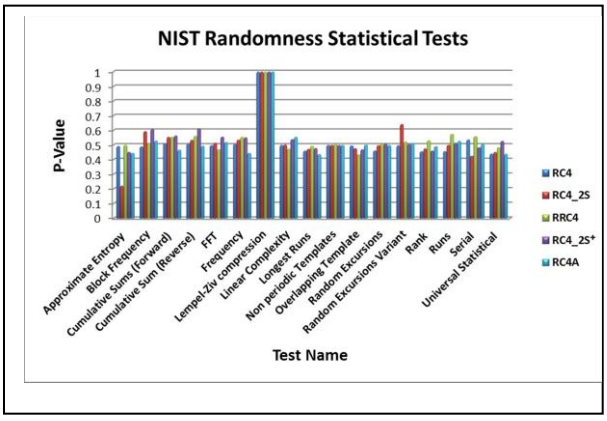

Figure 9. NIST Statistical Testfor Algorithms with RC4A

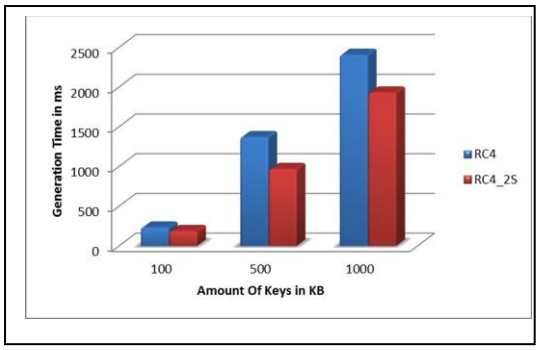

Figure 10. Key Generation Time for RC4 and RC4-2S

8. CONCLUSION

Many security applications use the stream cipher for data encryption, and the robustness of the cipher depends on the strength of the key stream being generated. The RC4 cipher system is an important encryption algorithm that can be used to protect the information on various communication networks as its implementation is simpler, and its cryptographic function is faster than that of AES. RC4 with a random initial state (RRC4) provides more randomness in the sequences, including its initial state and is more secure than RC4. The RC4 stream cipher with two state tables (RC4-2S) offers an enhanced randomness in the generated key sequences while the key generation time of RC4-2S is faster than that of RC4. RC4 stream cipher with two state tables to generate four key in each cycle (RC4-2S^{+}) achieves improved randomness in the generated key sequences without impacting its speed. The generated output sequences of all proposed algorithms have passed the NIST suite of statistical tests. The suggested algorithms are not complex at all, and can be implemented in either software or hardware.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grant CRI CNS-0855248, Grant EPS-0701890, Grant EPS-0918970, and Grant MRI CNS-0619069.

REFERENCES

[1] Sharma, K., Ghose, M.K., Kumar, D., Singh, R.P.K., Pandey V.K.: A Comparative Study of Various Security Approaches Used in Wireless Sensor Networks.In: *International Journal of Advanced Science and Technology*, vol. 177, no. 77, Apr. 2010.

[2] Wu, Y., Ma, D., Li, T., Deng, R.H.: Classify Encrypted Data in Wireless Sensor Networks. In:*Proc. IEEE 60th Vehicular TechnologyConf.*, pp. 3236-3239,2004.

[3] Turan, M.S, Doganaksoy, A., Calik, C.: Statistical Analysis of Synchronous Stream Ciphers. In*: SASC 2006: Stream Ciphers Revisited*,2006.

[4] Ahmad, S., Beg, M.R., Abbas, Q., Ahmad, J., Atif, S.: Comparative Study between Stream Cipher and Block Cipher using RC4 and Hill Cipher. In: *International Journal of Computer Applications (0975 - 8887)*, vol. 1, no. 25, 2010.

[5] Bakhtiari, M., Maarof, M.A.:An Efficient Stream Cipher Algorithm for Data Encryption. In:*IJCSI International Journal of Computer Science Issues*, Vol. 8, Issue 3, No. 1, May 2011.

[6] Menezes, A.J., Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography.In:*CRC Press*, Boston, 2001.

[7] Gupta, S.S., Chattopadhyay, A., Sinha, K., Maitra, S., Sinha, B.: High-Performance Hardware Implementation for RC4 Stream Cipher. In: *Computers, IEEE Transactions*, vol.62, no.4, pp.730 -743, April 2013.

[8] Zoltak, B.: VMPC One-way Function and Stream Cipher.In: *Fast Software Encryption. 11th International Workshop*,FSE' 2004, Springer-Verlag, pp. 210–225, 2004.

[9] Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the keyScheduling Algorithm of RC4. *Proceedings of Annual Workshop on Selected Areas in Cryptography*, vol. 2259, Toronto, Canada, pp. 1-24, Springer, 2001.

[10] Paul,S.,Preneel, B.: A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher.In:*Fast Software Encryption*, FSE 2004, pp. 245–259, Springer-Verlag, 2004.

[11] Maximov, A.: Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of the RC4 Family of Stream Ciphers. In: *Fast Software Encryption*, FSE 2005, Vol 3557, pp 342-358, Springer, 2005.

[12] Yu, Q., Zhang,C.: RC4 State and Its Applications. In: *Ninth Annual International Conference on Privacy, Security, and Trust*, pp.264-269, 2011.

[13] Pardeep, Pateriya, P.: PC-RC4 Algorithm: An Enhancement Over Standard RC4 Algorithm. In:*International Journal of Computer Science and Network (IJCSN)*,Vol. 1, Issue 3, June 2012.

[14] Kamble, B.H., Meshram, B.B.: Robustness of RC4 Against DifferentialAttack. In: *International Journal of Advanced Research in Computer Engineering & Technology*, Volume 1, Issue 4, ISSN: 2278 -1323, June 2012.

[15] Kadry,S.,Smaili, M.: An Improvement of RC4 Cipher Using Vigenère Cipher. In:*International Journal of Computational Intelligence and Information Security*, Vol. 1, No. 3, May 2010.

[16] Mironov, I.: (Not So) Random Shuffles of RC4. In*: Advances in Cryptology- CRYPTO 2002, 22nd Annual International Cryptology Conference*, vol. 2442 of Lecture Notes in Computer Science, pp. 304-319,Springer, 2002.

[17] Mousa, A., Hamad, A.: Evaluation of the RC4 Algorithm for Data Encryption. In: *International Journal of Computer Science & Application*, Vol. 3, No. 2, June 2006.

[18] Yao,Y., Chong,J., Xingwei,W.: Enhancing RC4 Algorithm for WLAN WEP Protocol. *In Control and Decision Conference (CCDC)*, pp.3623-3627, IEEE, 2010.

[19] Chefranov, A.G., Mazurova, T.A.: Pseudo-Random Number Generator RC4 Period Improvement. In: *Automation, Quality and Testing, Robotics, 2006 IEEE International Conference*, May 2006.

[20] Hammood, M.M.,Yoshigoe, K., Sagheer, A.M.: RC4 Stream Cipher with A Random Initial State. *Proceedings in 10th FTRA International Conference on Secure and Trust Computing, data management, and Applications (STA'13), Lecture Notes in Electrical Engineering*, Springer, 2013.

[21] Hammood, M.M.,Yoshigoe, K., Sagheer, A.M.: RC4-2S: RC4 Stream Ciphers with Two State Tables. *Proceedings in 5th FTRA International Conference on Information Technology Convergence and Services (ITCS'13),Lecture Notes in Electrical Engineering*, Springer, 2013.

[22] Grosul,A., Wallach,D.:ARelated-Key Cryptanalysis of RC4.*Department of Computer Science, Rice University, Technical Report TR - 00-358*, June 2000.

[23] Stamp,M.: Information Security Principles and Practice.*John Wiley & Sons, Inc.*, 2006.

[24] Sharif, S.O., Mansoor, S.P.: Performance Analysisof Stream and Block Cipher Algorithms. In: *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol.1, pp 522-525 IEEE, 2010.

[25] Mantin,I., Shamir,A.: A Practical Attack on Broadcast RC4.In: *8th International Workshop, FSE*,pp152-164, 2001.

[26] Roos, A.: A Class of Weak Keys in the RC4 Stream Cipher. In: *Vironix Soft-ware Laboratories, Westville, South Africa* Sep. 1995.

[27] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.*NIST special publication 800-22, National Institute of Standards and Technology (NIST)*, Gaithersburg, MD, USA (2001); See http://csrc.nist.gov/rng/ .

[28] Stallings,W.: Cryptography and Network Security Principles and Practices.*Prentice Hall, Fifth Edition, Pearson*, 2011.

[29] Mohamed,E. M.,El-Etriby, S.,Abdul-Kadar, H. S.: Randomness Testing of Modern Encryption Techniques in Cloud Environment.*8thInternational Conference on INFOrmatics System*, 2012.

[30] Singhal, N.,Raina, J. P. S.:Comparative Analysis of AES and RC4 Algorithms for Better Utilization.*International Journal of Computer Trends and Technology*,ISSN: 2231-280, July to Aug Issue 2011, pp. 177-18, 2011.

[31] Maitra,S.,Paul, G.: Analysis of RC4 and Proposal of Additional Layers for Better Security Margin. *Proc. Indocrypt*, vol. 5362,of LNCS,pp 27-39, 2008.

[32] Ekdahl, P.,Johansson, T.: SNOW - A New Stream Cipher*. *Proceedings of first NESSIE Workshop*, Heverlee, Belgium, 2000.

[33] Rogaway,P.,Coppersmith,D.: A Software Optimized Encryption Algorithm.Journal of Cryptology, 11(4):273-287, 1998

[34] Hawkes, P., Rose, G.: Primitive Specification and SupportingDocumentationfor SOBER-t32 Submission to NESSIE. *Proceedings of the first NESSIE Workshop*, Heverlee, Belgium, 2000.

[35] Wagner,D., Schneier,B.,Kelsey,J.: Cryptanalysis of ORYX. In: *Proc.5th Annu. Workshop Selected Areas in Cryptography (WSK)*, pp. 296 - 305, 1998.

[36] McGrew,D., Fluhrer, S.: The Stream Cipher Leviathan. *NESSIE projectsubmission*, October 2000.

[37] Lai, X., Massey, J.:A Proposal for a New Block Encryption Standard.In: *Proc. Eurocrypt '90*, volume 473 of LNCS,pp 389-404, Berlin,Springer,1991.

[38] Watanabe, D. Furuya, S., Yoshida, H., Preneel,B.: A New Keystream Generator MUGI. In: *Fast Software Encryption (FSE) 2002, Lecture Notes in Computer Science*, vol. 2365, pp179-194 Springer, 2002.

[39] Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., Kohno, T., Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive, In: *Johansson, T. (ed.) FSE 2003. LNCS*, vol. 2887, pp. 330–346. Springer, Heidelberg, 2003.

[40] Galanis, M.D., Kitsos, P., Kostopoulos, G., Sklavos, N., Koufopavlou, O.,Goutis, C.E.: Comparison of the hardwareArchitectures and FPGA Implementations of Stream Ciphers.In:*Proceedings of the 2004 11th IEEE International Conferenceon Electronics, Circuits and Systems (ICECS'04)*, pp.571 - 574, 2004.

Maytham M. Hammood received his B.Sc. and M.Sc. in Computer Science from the University of Technology, Baghdad, Iraq in 2002 and 2005 respectively. He received M.S. in Applied Science from the University of Arkansas at Little Rock, USA, December 2013.He is presently a Ph.D. candidate of Computer Science, the University of Arkansas at Little, USA. His research areas of interest are of data protection mechanisms for wireless sensor network and security algorithms enhancements.

Kenji Yoshigoeis an Associate Professor in the Department of Computer Science and the Director of Computational Research Center (CRC) at UALR. He received his Ph.D. degree in Computer Science and Engineering from the University of South Florida. He is currently investigating the reliability, security, and scalability of various interconnected systems ranging from tightly coupled high performance computing systems to resource-constrained wireless sensor networks.

Ali M. Sagheer is an Associate 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.