Journal of AL-Qadisiyah for computer science and mathematics Vol.10 No.3 Year 2018
ISSN (Print): 2074 – 0204 ISSN (Online): 2521 – 3504

Design of keystream Generator utilizing Firefly Algorithm

Mohammed Salih Mahdi
University of Information Technology and Communications
BIT Dept., Business Information College
Baghdad, Iraq
mohammed.salih@uoitc.edu.iq

Nidaa Flaih Hassan
University of Technology
Computer science Dept
Baghdad, Iraq
110020@uotechnology.edu.iq

Recived : 1\8\2018 **Revised : 6\9\2018** **Accepted : 12\9\2018**

Available online : 26/9/2018

DOI: 10.29304/jqcm.2018.10.3.441

Abstract :

Stream cipher is one of encryption procedures for sending data in internet; stream cipher is suitable in telecommunications and real-time apps. The robustness measurement of stream cipher is according to the randomness of keystream that is utilized. If the random series of keystream generator is low, the keystream of stream cipher can be read and encrypted data by stream cipher become vulnerable to attackers. This paper utilizes Firefly Algorithm based Local Key Generation for generation keystream. The generated keystream is independent of original messages. The randomness of keystream series of Firefly passing the five standard criteria. The suggested keystream generator is word-established appropriated to fast real-time apps than are bit-established linear stream ciphers. Furthermore, the suggested keystream generator satisfies the three demands of benchmarks such as maximum correlation, robust randomness and huge complexity.

Key Words. Stream Cipher, keystream Generator, Firefly Algorithm, Local Keys Function**.**

1. Introduction

Cryptography is an exercise together with analyze procedures for safe communication in the presence of adversaries. Generally, it is about founding and studying protocols which overcome the technique of attackers and that are linked to a diversity of parts in data security [1]. The mammoth issues of secure telecommunication is keeping the confidential information from interception. In cipher frameworks , It is popular that the experts of encryption procedures required techniques to discover an orderly procedure in checking their ciphers to guarantee that they are protected from attackers [2][3]. One of a popular symmetric procedure is stream cipher, all portion of original message and keystream are encrypted together. The keys of Stream cipher utilized for encryption procedure is altered randomly. Consequently, the cipher that's created is mathematically very hard to breach. .The altering of random keystream will not permit any sample to be frequent that allocate a guide to attacker to breach cipher data [4], [5].

Stream cipher is suitable on equipment and programs, and in some situations obligatory in telecommunications and real-time apps chiefly with restricted memory [6], [7]. Stream cipher is less apt to cryptanalysis due to identical portions of original messages are encipher with various portions of the keystream [8]. The essential idea of stream cipher is one-time pad cipher which is refer to Vernam cipher, necessity that is true random series with generated keystream , Both transmitter and recipient are participated keystream , and keystream only utilized once [9]. The cons of stream cipher are the total volume of the keystream and the original message should be the identical. Subsequently a huge quantity of keystream have to be kept and transmitted. Furthermore, if the random series is discover , the keystream utilized for encryption procedure can be keep track readily [10].

Many different papers utilized to evolve Stream Cipher generator for Encryption according to SI and several techniques, In[11] suggested keystream procedure for encryption data according to Ant Colony and the allocation of letters in the original message. In [12] suggested keystream generator according to a Particle swarm optimization for encryption data. The con of these articles is that enumeration the appearance of letters of keystream in the original messages. Of course, that it excess the overall time consumption for encryption, when the volume of keystream selected is huge.

In [13] suggested keystream generator according to 3D chaotic maps and Particle swarm optimization to create a random number generator and tested by five standard criteria. In [14] suggested keystream generator according to on dynamic the thought of updates composite LFSR stream cipher every update in message that indicates to has a huge complexity encryption procedure and tested by standard criteria

.

Concerning , the stream cipher and Firefly Algorithm that is debated at the previously, the contribution of this paper is suggested Keystream generator utilizing Firefly Algorithm based Local Key Generation where the generated keystream is independent of original messages .

2. Firefly Algorithm

Optimization is an arithmetic procedure for getting either highest or smallest value of a target function by selecting a best solution from a set of solutions [15]. There are a huge number of optimization problems, which were complicated and toughed to discover a solution at a sensible time in different areas like business, operations research and computer science [16].

These optimization problems can be successfully solved by applied Biology-Nature-Inspired-Metaheuristic-Algorithms (BNIMAs) mostly that according to swarm intelligence (SI) which is a subset of Artificial Intelligence. Birds, Wasps, Ants, Bugs ,Bees and Firefly are different models of the family of BNIMAs which are using the behavior of SI based on target function [17][18].

In optimization problems, firefly algorithm is a one of SI clans of algorithms that lately exhibited magnificent performances by presenting best solutions [19] as shown in Algorithm (1). Firefly algorithm utilizes the subsequent three essentials [20][21]:

- All firefly would be going to other fireflies that have a high attractiveness irrespective of their gender.
- The attraction of a firefly is apt to its lighting that is minify as distance of different firefly rises. Fireflies shall move arbitrarily, when no it's lighter one than a specific firefly.
- The aim of target function is locating the illumination of firefly.

Three critical criteria in Firefly algorithm can be summarized as [22]:

Journal of AL-Qadisiyah for computer science and mathematics Vol.10 No.3 Year 2018
ISSN (Print): 2074 – 0204 ISSN (Online): 2521 – 3504

- The 1st critical criteria is attraction (attractiveness N) that is specified by its lighting strength as Eq.1 follows:

$N=N_0 e^{-vt^2}$ Eq.1

N_0 is an attraction at distance t equal to zero, while v is refer to lighting absorption in weather.

- The 2nd critical criteria is distance that is specified by utilizing the Cartesian distance between two fireflies as Eq.2 follows:

$R_{i,j} = \sqrt{\sum_{s=1}^{n} (F_{I,S} - F_{J,S})^2}$ Eq.2

In the n-dimensional space, $F_{i,s}$ is s-th component of coordinate F_i of i-th Firefly.

- The 3rd critical criteria is movement that is specified by brightness for instance, firefly (a) would be going to firefly (b) that has a high attractiveness as Eq.3 follows:

$M_a=M_a + N_0 e^{-vR^2 ij} (M_b - M_a) + P (r-0.4)$ Eq.3

Where P is refer to randomization value, r is refer to a random value that its range between 0 and 1 a, M_a is refer to movement firefly (a) and M_b is refer to movement firefly (b). Furthermore Firefly is equal to particle swarm optimization when v is equal to zero [22].

Algorithm (1): Firefly Algorithm
Input: number of firefly population ,light absorption coefficient v,max- iteration, objective
Output: set of keystream
Begin $Step_1$: Determine the objective function. $Step_2$: Generate initial population of fireflies. $Step_3$: Calculate the Light intensity at fireflies based on objective function. $Step_4$: iteration=1 $Step_5$: A=1,B=1 $Step_6$: if fitness function of firefly (A) less than fitness function of firefly (B) then Move firefly (A) towards firefly (B) $Step_7$: modify attraction differs with distance through e^{-vt^2} $Step_8$: Evaluate modern resolutions and modify brightness intensity. $Step_9$:B=B+1 $Step_{10}$: if (B less than or equal to number of fireflies) Goto $Step_6$ $Step_{11}$: A=A+1, B=1 $Step_{12}$: if (A less than or equal to number of fireflies) Goto $Step_6$ $Step_{13}$: Rank fireflies and find the current best $Step_{14}$: iteration= iteration +1 $Step_{15}$: if (iteration less than or equal to max-iteration) Goto $Step_5$ **End**

3. Methodology of Designing Stream Cipher utilizing Firefly Algorithm

The main essentials of Firefly Algorithm and stream ciphers have been learned and analyzed. In this Methodology, Firefly Algorithm based Local Key Generation (FAbLKG) is suggested for generation keystreams as exhibited in Algorithm (2). In FAbLKG, a Firefly is utilized to assign a keystream. Each Firefly have many keys inside that keystream, the keys in the keystream are set of bits can be 0 or 1. For instance, if the length of keys of each Firefly is equal to 512 consequently it is depicted by Eq.4.

$\text{Firefly (keystream)} \ key1, key2 , . , key512$ Eq. 4

Algorithm (2): Firefly Algorithm based Local Key Generation
Input: number of firefly population , max-iteration
Output: set of keystream
Begin $Step_1$: iteration=1 $Step_2$: For each firefly in the population, randomly generate the initial keystream utilizing the Local Key Generation. $Step_3$: Calculate the Light intensity based on fitness function of keystream of each firefly $Step_4$: A=1,B=1 $Step_5$: if fitness function of firefly (A) less than fitness function of firefly (B) then $Step_{5.1}$: Compute hamming distance between firefly (A) and firefly (B) $Step_{5.2}$: Determine number of swap of keys in the keystream of firefly (A) between 1 and Hamming Distance $Step_{5.3}$: Making swap operation on different locations of keys in the keystream of firefly (A) according to the possible range of $Step_{5.2}$. $Step_{5.4}$: update fitness function of keystream of firefly (A) and modify brightness intensity. $Step_6$: B=B+1 $Step_7$: if (B less than or equal to number of fireflies) Goto step5 $Step_8$:A=A+1, B=1 $Step_9$: if (A less than or equal to number of fireflies) Goto step5 $Step_{10}$: Rank fireflies and check fitness function of them, if its equal to five then store the keystream obtained. $Step_{11}$: iteration= iteration +1 $Step_{12}$: if (iteration less than or equal to max-iteration) Goto step4 **End**

Journal of AL-Qadisiyah for computer science and mathematics Vol.10 No.3 Year 2018
ISSN (Print): 2074 – 0204 ISSN (Online): 2521 – 3504

3.1 Representation of Fireflies

For generating a keystream, Firefly Algorithm utilize the binary code as a solution. The Firefly Algorithm initiates with an elementary population encompasses set of fireflies (keystream) according to Local Key Generation.

3.1.1 Local Key Generation

Local Key Generation **(LKG)** is a novel word-established NLFSR stream ciphers which is offer numerous volumes of keystream per round than bit-established LFSRs stream ciphers, according to the word volume. Thus, in LKG an equilibrium is accomplished between security and efficiency. LKG can be referred as a development of the outcome feedback mode. The outcome of keystream is furthermore the feedback to the inner state. LKG has initially seed of 512 bits inner state sets by initialization vector. These bits are splitted into sixteen 32-bit words categorized W_1 to W_{16} and making XOR and Local Function feeding among some of them as clarified in Figure (1).

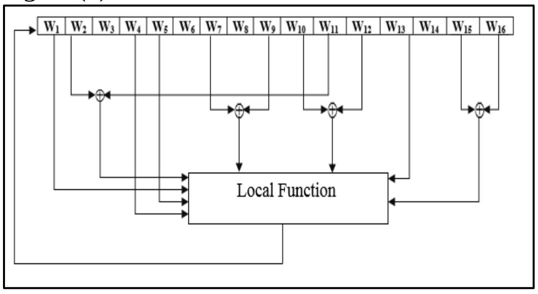

Figure (1): Local Key Generation

The Local Function that is called on one occasion per iteration, impacts the inner state to produce 512 bits of keystream per iteration. The input of Local Function is 512 bits (eight variable, the size of each variable is 32 bits). In Local Function, A network of summation mod 2^{32} and XOR are utilized for diffusion and it conclude of three procedures as clarified in algorithm (3). The 1^{st} procedure of Local Function is Pre-confusion. The 2^{nd} procedure of Local Function is combination (M and Q Functions) according to two s-box ($O_1$and O_2)as clarified in algorithm (4). The 3^{rd} procedure of Local Function is Post-confusion.

Algorithm (3): Local key Function

Input: Sixteen 32-bit word (W1 to W16), n=number of keystream

Output: 512- bit (keystream)

Begin

Step1: iteration=1

Step2: // Map initialization

$R_1=W_1$ $R_2=W_7 \oplus W_9$ $R_3=W_4$

$R_4=W_{15} \oplus W_{16}$ $R_5=W_5$ $R_6=W_{13}$ $R_7=W_{10} \oplus W_{12}$ $R_8=W_2 \oplus W_{11}$

Step3: // Pre-confusion. 2^{32}= 4294967296

$R_1 = (R_1 + R_8) \text{ Mod } 2^{32}$ $R_2=R_2 \oplus R_7$

$R_3 = (R_3 + R_6) \text{ Mod } 2^{32}$ $R_4=R_4 \oplus R_5$

$R_5 = (R_5 + R_1) \text{ Mod } 2^{32}$ $R_6=R_6 \oplus R_2$

$R_7= (R_7 + R_3) \text{ Mod } 2^{32}$ $R_8=R_8 \oplus R_4$

Step4:// combination functions = M_1, M_2, M_3,M_4,Q_1, Q_2 ,Q_3 Q_4

$R_1 = R_1 \oplus M_1(R_2)$ $R_2 = R_2 \oplus Q_3(R_5)$

$R_3 = R_3 \oplus M_2(R_8)$ $R_4 = R_4 \oplus Q_4(R_6)$

$R_5 = R_5 \oplus M_3(R_3)$ $R_6 = R_6 \oplus Q_1(R_7)$

$R_7 = R_7 \oplus M_4(R_4)$ $R_8= R_8 \oplus Q_2(R_1)$

Step5: // Post-confusion

$R_1=R_1 \oplus R_6$ $R_2 = (R_2 + R_4) \text{ Mod } 2^{32}$

$R_3=R_3 \oplus R_5$ $R_4 = (R_4 + R_8) \text{ Mod } 2^{32}$

$R_5=R_5 \oplus R_2$ $R_6= (R_6 + R_4) \text{ Mod } 2^{32}$

$R_7=R_7 \oplus R_1$ $R_8= (R_8 + R_6) \text{ Mod } 2^{32}$

Step6: //Update inner state

$W_i= R_i$ $1 \leq i \leq 8$

$W_i= R_{i-8}$ of $Step_2$ $9 \leq i \leq 16$

Store the update sixteen 32-bit words called W as keystream

iteration = iteration +1

Step6: if (iteration less than or equal to n) Goto $Step_2$

End

Algorithm (4): M and Q Functions

Input: 32-bit , N=number of M and Q function,O1= 32 × 32 and O2= 32 × 32

Output: Update 32-bit

Begin

Step1: Break 32 –bit to four variable (y_1,y_2,y_3,y_4)

Step2:

if (N==1) then

$M_1(y) = O_1(y_1) \oplus O_1(y_2) \oplus O_1(y_3) \oplus O_2(y_4)$

$Q_1(y) = O_2(y_1) \oplus O_2(y_2) \oplus O_2(y_3) \oplus O_1(y_4)$

Else if (N==2) then

$M_2(y) = O_1(y_1) \oplus O_1(y_2) \oplus O_2(y_3) \oplus O_1(y_4)$

$Q_2(y) = O_2(y_1) \oplus O_2(y_2) \oplus O_1(y_3) \oplus O_2(y_4)$

Else if (N==3) then

$M_3(y) = O_1(y_1) \oplus O_2(y_2) \oplus O_1(y_3) \oplus O_1(y_4)$

$Q_3(y) = O_2(y_1) \oplus O_1(y_2) \oplus O_2(y_3) \oplus O_2(y_4)$

Else

$M_4(y) = O_2(y_1) \oplus O_1(y_2) \oplus O_1(y_3) \oplus O_1(y_4)$

$Q_4(y) = O_1(y_1) \oplus O_2(y_2) \oplus O_2(y_3) \oplus O_2(y_4)$

End if

End

3.2 Fitness Function of Fireflies

For each firefly, the fitness function is utilize to measure the robustness of keystream as brightness (Light Intensity) which is computed by testing the five standard criteria of generated keystream as interpreted in Table (1). If the generated keystream is exceeding all five standard criteria therefore robustness of Firefly (keystream) is five and if the generated keystream is exceeding four of standard criteria therefore robustness of Firefly is four and so on.

3.3 Moving of Fireflies

The Firefly Algorithm initiates with an elementary population randomly according to Local Key Generation consequently the fireflies dispersed in state space. If the robustness of keystream (lighting) of Firefly (A) less than the firefly(B) consequently Firefly (A) move in the direction of the Firefly (B) by swapping the keys in the keystream of the Firefly (A). The range of swapping keys in the keystream of Firefly (A) is determined randomly from one to Hamming Distance between Firefly (A) and Firefly (B) as depicted by Eq2. Indeed, the swapping keys in the keystream creates modern diffusions (permutations). For example, assume the fitness (robustness of keystream) of Firefly (A) and Firefly (B) are 2 and 4 respectively, and the Hamming Distance between Firefly (A) and Firefly (B) is 5. Consequently, Firefly (A) move in the direction of the Firefly (B) by swapping keys in the keystream of the Firefly (A). The range of swapping keys in the keystream of Firefly (A) is randomly selected between 1 and 5 (Hamming Distance). Assume the selected the range of swap is 3 that is led to making at most three swap of keys (different swap locations) in the keystream of Firefly (A).

Table (1): Five Standard Criteria Equations Information [23]

Five Standard Criteria Equations	**Information on Five Standard Criteria**
$T1 = \frac{(M0 - M1)^2}{M}$	M0: number of 0's in keystream. M1: number of 1's in keystream. M: total size of keystream.
$T2 = \frac{4}{M-1}((M11)^2 + (M00)^2 + (M01)^2 + (M10)^2) - \frac{2}{M}(M1^2 + M0^2) + 1$	M11: number of 11's in keystream. M00: number of 00's in keystream. M01: number of 01's in keystream. M10: number of 10's in keystream
$T3 = \frac{2^N}{P}\left(\sum_{j=1}^{2^N} {M_j}^2\right) - P$	M_j: number of appearance of the j[th] of length N $P = \frac{M}{N}$ $\frac{M}{N} \geq (5 * 2^N)$
$T4 = \left(\sum_{j=1}^{N} \frac{(B_j - P_j)^2}{P_j}\right) + \left(\sum_{j=1}^{N} \frac{(G_j - P_j)^2}{P_j}\right)$	N :maximum j for which $P_j \geq 5$. B_j: Amount of blocks (subsequences runs of 1's) of length j in M. G_j: amount of gabs (subsequences runs of 0's) of length j in M. $P_j = \frac{M - j + 3}{2^{(j+2)}}$
$T5 = \frac{2\left(A(k) - \frac{(M-k)}{2}\right)}{\sqrt{(M-k)}}$	k : $1 \leq k \leq [m/2]$ $A(k) = \sum_{j=0}^{M-k-1} (S_j + S_{+k}) \text{ Mod } 2$

4. Case Study of Suggested Keystream Generator

Suppose, the number of firefly population is two and max- iteration is one. The following steps illustrated the suggested keystream generator according to Algorithm (2), Algorithm (3) and Algorithm (4).

- S_1: Suppose the seed of two fireflies:

Firefly(A)→"5d413d691dd67b3449bf371ac15968af
eebe6361f83cf5c359ae7fc83178ab8ce0157eba9a3
8655da4abe24f359f664ad6036972c002764f33738
94d49a6df77"

Firefly(B)→"27d591c45f02e8c6d0624a6a20d9b4bb
5f59a6e8e497a2e235feb4ad5d45ec8aca348c8f24a
269ae9c608d099a652cd973f9098e69360fab3c318
80567ca82e4".

Each one of fireflies is represented by 128 hexa-number that is equal to 512 bits.

- S_2: At first, each firefly is splitted to sixteen 8-hexa-number (W_1 to W_{16}) ,for instance Firefly(A) is splitted to (W_1= "5d413d69", W_2="1dd67b34",…,W_{16}= "49a6df77").
- S_3: For Map initialization of keystream for Local key Function utilized according to **S**tep$_2$ of Algorithm (3), $R_1= W_1$→ R_1= "5d413d69" → $R_2= W_7 \oplus W_9$ → R_2="59ae7fc8" ⊕ "e0157eba"="b9bb0172" and so.
- S_4: For Pre-confusion of keystream for Local key Function utilized according to **S**tep$_3$ of Algorithm (3), $R_1 = (R_1 + R_8)$ Mod 2^{32} →R_1="16bed6e4", $R_2=R_2 \oplus R_7$ → R2="161c0265"and so.
- S_5: For combination functions of keystream for Local key Function utilized according to **S**tep$_4$ of Algorithm (3), $R_1 = R_1 \oplus M_1(R_2)$ At first, calculating M_1 (R_2) according to Algorithm (4),y=R_2. y will be splitted to y1,y2,y3and y4, considering as indexing of O_1; O_2 (two S-box in Appendix) so, $M_1(y) = O_1(y_1) \oplus O_1(y_2) \oplus O_1(y_3) \oplus O_2(y_4)$ → $M_1(R_2)$="5ddc0340" then make ⊕ with "16bed6e4"→ R_1 = "5bc6f670" and so on.
- S_6: For Post-confusion of keystream for Local key Function utilized according to **S**tep$_5$ of Algorithm (3),

$R_1=R_1 \oplus R_6$ → R_1="b80d61b4" ⊕ "e3cb97c4" →R_1="b80d61b4" and so on.

- S_7: For Update inner state of keystream for Local key Function utilized according to **S**tep$_6$ of Algorithm (3), $W_i= R_i$, $1 \leq i \leq 8$, $W_i= R_{i-8}$ of **S**tep$_2$ of Algorithm (3), $9 \leq i \leq 16$, W1="b80d61b4",W2="a71ed0fa", and so on.
- S_8: The output of local key function of two fireflies:
  Firefly(A)→="b80d61b4a71ed0fa97c5bc4c2
  02b6aebfa54589b03f702af5cf8116440967f3
  65d413d691dd67b3449bf371ac15968afeebe
  6361f83cf5c359ae7fc83178ab8c"
  Firefly(B)→"b94b365494d6e6db29b3a3d6ef
  a21ef6cda2534b5e42a326606cba30407549e
  227d591c45f02e8c6d0624a6a20d9b4bb5f59
  a6e8e497a2e235feb4ad5d45ec8a".
- S_9: for each firefly, calculating Light intensity According to five standard criteria that is illustrated in Table (1). The fitness function of firefly (A) is 5, i.e. firefly (A) is passing five standard criteria. However, the fitness function of firefly (B) is 3,i.e. firefly (B) is failing in serial test and run test.
- S_{10}: According to Step$_5$ of Algorithm (2), compute hamming distance between firefly (A) and firefly (B) by convert them to binary and count the matching number as maximum number of making swap operation on different locations.
- S_{11}: check fitness function of two firefly, already, firefly (A) is passing five standard criteria early then discovering firefly (B) is also passing five standard criteria too due to iterated swap processing. So the final result of two fireflies in binary form.
  firefly(A)="101110000000011010110000110
  1101001010011100011110110100001111110
  1010010111110001011011110001001100000
  1000000010101101101010111010111111110
  1001010100010110001001101100000001111
  1101110000001010101111010111001111110
  0000010001011001000100000001001011001
  1111110011011001011101010000001001111
  0101101001000111011101011001111101100
  1101000100100110111111001101110001 10
  1011000001010110010110100010101111111
  1011101011111001100011011000011111110
  0000111100111101011100001101011001 10
  1011100111111111001000001100010111110
  0010101011110001100".
  Firefly(B)="101111000100101110010110 01
  1101000001010011010110111000110101 10
  1100101001101101101010001101010100 11
  0011111010001001011111111101101100 10
  01

١٠٠٠١٠١٠٠٠٠١٠٠١١٠١٠٠١٠١١٠١٠٠١١١٠٠٠
٠١٠١١٠١٠١١٠٠١١٠٠٠٠٠١١٠٠١٠٠٠٠٠٠٠١١٠
١٠٠٠١٠١١١٠١٠٠١١١٠٠٠٠٠١٠٠٠١٠٠٠١١١٠١
٠١٠١٠٠١٠٠١١١١٠٠٠١٠٠٠١٠١٠١١١١١٠٠١١١
١٠٠١٠٠٠١١٠٠٠٠٠١٠١٠١٠١١١٠١٠٠٠٠٠٠١٠١١
١٠١٠٠٠١١١٠٠١١٠١١٠١٠٠٠٠٠١١٠٠٠١١٠١١٠
١٠١٠١٠١٠١٠١١٠٠١١٠٠٠٠١١٠١١٠٠١٠٠١١١٠١
٠٠١٠١١١٠١١١١٠١١١١٠٠١١١١٠٠٠١٠٠٠١١١٠
١١١٠١٠٠٠٠١١١001001001011110100010111
0001000110101111111110101101001010110
1010111010100010111101100100010 10" .

The implementation of above case study of suggested keystream Generator for generating two fireflies as two Keystream as showing in Figure (2) utilizing JavaScript language

Figure (2): Implementation of sample Case Study

5. Results of Suggested Keystream Generator

This paper utilizing one of SI procedures called firefly with local key generation for generation keystream. Through implementation of suggested key generator, the procedure has several variables as illustrates in Table 2.

Table 2. Variables chosen of suggested keystream generator

Variables	Range
number of firefly population	2 to 1000
max- iteration	1 to 1000
Length of firefly	512 bits
Number of swap of keys in the keystream of firefly.	1 to Hamming Distance

If utilize only firefly algorithm for generation keys will be stuck in local optimal and some of generated keystream will be poor. While, the generated keystream of suggested FAbLKG has high randomness by passing the five-benchmark tests due to utilizing local key generation and firefly algorithm. Local key generation is prevent the firefly algorithm from stuck in local optimal and generated keystream

The suggested FAbLKG is implemented on various entire size of data utilizing JavaScript language. The FAbLKG performance is utilized on many keystream series of Firefly of length 512 bits. Each iteration, discovering the best keystream if it found then store it, when the robustness of keystream series of Firefly is five that means passing the five benchmark tests.
For instant, the result of one keystream series of Firefly that is passing the five standard criteria as interpreted in Table (3).

Table (3): Five Standard Criteria Performance

5 - benchmark Tests	Test Value	Threshold	Test value< Threshold
Frequency T1	0.007	3.841	pass
Serial T2	4.941	5.991	pass
Poker T3	11.813	24.995	pass
Runs T4	5.028	12.591	pass
Autocorrelation T5	0.531	1.96	pass

5. Conclusion

This paper suggests FAbLKG with robust infrastructure with more diffusion of the generated keystream, From examining the best keystream of rounds by utilizing FAbLKG and rely on randomness standard, it is simple to observe that the suggested FAbLKG have a huge keystreams solutions which are satisfied the three demands of benchmarks such as maximum correlation, robust randomness, huge complexity. Attacker want to 2^{512} prospective trails to breach of the generated keystream of FAbLKG, consequently in this situation, a brute-force attacking seems unwieldy step. FAbLKG is Word-established may be better appropriated to fast real-time apps than are bit-established linear stream ciphers, FAbLKG is offer numerous volumes of keystream per iteration than bit-established LFSRs.

References

[1] H. Feistel, “Cryptography and computer privacy,” *Sci. Am.*, vol. 228, no. 5, pp. 15–23, 1973.

[2] H. Gustafson, E. Dawson, L. Nielsen, and W. Caelli, “A computer package for measuring the strength of encryption algorithms,” *Comput. Secur.*, vol. 13, no. 8, pp. 687–697, 1994.

[3] G. J. Krishna, R. Vadlamani, and S. N. Bhattu, “Key Generation for Plain Text in Stream Cipher via Bi-Objective Evolutionary Computing,” *Appl. Soft Comput.*, 2018.

[4] P. P. Deepthi, D. S. John, and P. S. Sathidevi, “Design and analysis of a highly secure stream cipher based on linear feedback shift register,” *Comput. Electr. Eng.*, vol. 35, no. 2, pp. 235–243, 2009.

[5] T. Good and M. Benaissa, “Hardware results for selected stream cipher candidates,” *State Art Stream Ciphers*, vol. 7, pp. 191–204, 2007.

[6] V. Menezes, “Oorschot:‘ Handbook of Applied Cryptography’, 1997,” *CRC Press*, vol. 200, pp. 6–10.

[7] M. Stamp, *Information security: principles and practice*. John Wiley & Sons, 2011.

[8] A. AL RASEDY and A. AL SWIDI, “An advantages and Dis Advantages of Block and Stream Cipher,” in *Third Annual Scientific of the Faculty of basic Education Conference.*, 2010.

[9] M. J. B. Robshaw, “Stream ciphers,” *RSA Lab. Tech. Rep.*, 1995.

[10] C. P. Pfleeger and S. L. Pfleeger, *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

[11] N. K. Sreelaja and G. A. V. Pai, “Swarm intelligence based key generation for text encryption in cellular networks,” in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, 2008, pp. 622–629.

[12] S. NK and G. A. V. Pai, “Design of Stream Cipher for Text Encryption using Particle Swarm Optimization based Key Generation., Journal of Information Assurance and Security ,30-41 2009

[13] R. A. Ali, “Random Number Generator based on Hybrid Algorithm between Particle Swarm Optimization (PSO) Algorithm and 3D-Chaotic System and its Application,” *Iraqi J. Inf. Technol.*, vol. 8, no. 3 pp. 1–20, 2018.

[14] A. A. Ghazi and F. H. Ali, “Robust and Efficient Dynamic Stream Cipher Cryptosystem,” *Iraqi J. Sci.*, vol. 59, no. 2C, pp. 1105–1114, 2018.

[15] H. C. Ong, S. L. Tilahun, and S. S. Tang, “A Comparative Study on Standard, Modified and Chaotic Firefly Algorithms,” *Goal Pertanika*, p. 251, 2015.

[16] C. Blum and X. Li, “Swarm intelligence in optimization,” in *Swarm Intelligence*, Springer, 2008, pp. 43–85.

[17] A. T. S. Al-Obaidi, H. S. Abdullah, and Z. O. Ahmed, “Meerkat Clan Algorithm: A New Swarm Intelligence Algorithm,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 10, no. 1, pp. 354–360, 2018.

[18] A. Slowik and H. Kwasnicka, “Nature Inspired Methods and Their Industry Applications—Swarm Intelligence Algorithms,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 3, pp. 1004–1015, 2018.

[19] I. Fister Jr, M. Perc, S. M. Kamal, and I. Fister, “A review of chaos-based firefly algorithms: perspectives and research challenges,” *Appl. Math. Comput.*, vol. 252, pp. 155–165, 2015.

[20] X.-S. Yang and X. He, “Firefly algorithm: recent advances and applications,” *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–50, 2013.

[21] H. Wang, Z. Cui, H. Sun, S. Rahnamayan, and X.-S. Yang, “Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism,” *Soft Comput.*, vol. 21, no. 18, pp. 5325–5339, 2017.

[22] X.-S. Yang, “Nature-Inspired Metaheuristic Algorithms,” *Luniver Press. UK*, pp. 242–246, 2008.

[23] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

Appendix A

S1 and S2 are generated randomly as follows:

S1=[0xB2C420F8,0x3218BD4E,0x84590D94,0xD51D3A8C,0xA3AB3D24,0x2A339E3D,0xFEE67A
23,0xAF844391,0x17465609,0xA99AD0A1,0x05CA597B,0x6024A656,0x0BF05203,0x8F559DDC,0
xEE171139,0x99B9E75F,0x8829A7ED,0x2C511CA9,0xD89BF75,0xF2F8CDD0,0x2DA2C498,0x48
314C42,0x922D9AF6,0xAA6CE00C,0xAC66E078,0x7D4CB0C0,0x5500C6E8,0x23E4576B,0x6B36
5D40,0x88C28D4A,0xCA6A8992,0xB40726AB,0x508C65BC,0xBE87B3B9,0x4A894942,0x9AEEC
C5B,0x6CA6F10B,0x303F8934,0xD7A8693A,0x7C8A16E4,0xB8CF0AC9,0xAD14B784,0x819FF9F
0,0x336BE860,0x5DBEEEFE,0x0E945776,0xD4D52CC4,0x0E9BB490,0x376EB6FD,0x6D891655,0
xD4078FEE,0xE07401E7,0xA1E4350C,0xABC78246,0x73409C02,0x24704A1F,0x478ABB2C,0xA0
849634,0x9E9E5FEB,0x77363D8D,0xD350BC21,0x876E1BB5,0xC8F55C9D,0xD112F39F,0xDF1A
0245,0x9711B3F0,0xA3534F64,0x42FB629E,0x15EAD26A,0xD1CFA296,0x7B445FEE,0xB8FC2E0
D,0x80168E15,0x0D7DEC9D,0xC5581F55,0xBE4A2783,0xD27012FE,0x53EA81CA,0xEBAA07D2
,0x54F5D41D,0xABB26FA6,0x41B9EAD9,0xA48174C7,0x1F3026F0,0xEFBADD8E,0x387E9014,0
x1505AB79,0xEADF0DF7,0x67755401,0xDA2EF962,0x41670B0E,0x0E8642F2,0xCE486070,0xA4
7D3312,0x4D7343A7,0xECDA58D0,0x1F79D536,0xD362576B,0x9D3A6023,0xC795A610,0xAE4D
F639,0x60C0B14E,0xC6DD8E02,0xBDE93F4E,0xB7C3B0FF,0x2BE6BCAD,0xE4B3FDFD,0x7989
7325,0x3038798B,0x08AE6353,0x7D1D20EB,0x3B208D21,0xD0D6D104,0xC5244327,0xF20DCDF
A,0xB7CB7159,0x58F3199F,0x9855E43B,0x1DF6C2D6,0x46114185,0xE46F5D0F,0xAAC70B5B,0
x48590537,0x0FD77B28,0x67D16C70,0x75AE53F4,0xF7BFECA1,0x6017B2D2,0xD8A0FA28,0x98
93F59F,0xE976832A,0xB1EB320B,0xA409D915,0x7EC6B543,0x66E54F98,0x5FF805DC,0x599B2
23F,0xAD78B682,0x2CF5C6E8,0x4FC71D63,0x08F8FED1,0x81C3C49A,0xE4D0A778,0xB5D369C
C,0x2DA336BE,0x76BC87CB,0x957A1878,0xFA136FBA,0x894A1911,0x909F21B4,0x6A7B63CE,
0xE28DD7E7,0x4178AA3D,0x4346A7AA,0xA1845E4C,0x166735F4,0x639CA159,0x58940419,0x4
E4F177A,0xD17959B2,0x12AA6FFD,0x1D39A8BE,0x7667F5AC,0xED0CE165,0xF1658FD8,0x28
B04E02,0x8F3C0E7B,0x7A1FF157,0x598324AE,0xFFBAAC22,0xD67DE9E6,0x3EB52897,0x4E07
E855,0x87CE73F5,0x8D046706,0xD42D18F2,0xE71B1727,0x38473B38,0xB37B24D5,0x381C6AE1
,0xE77D6589,0x6018CBFF,0x93CF3752,0x9B6EA235,0x504A50E8,0x464EA180,0x86AFBE5E,0xC
C2D6AB0,0xAB91707B,0x1DB4D579,0xF9FAFD24,0x2B28CC54,0xCDCFD6B3,0x68A30978,0x4
3A6DFD7,0xC81DD98E,0xA6C2FD31,0x0FD07543,0xAFB400CC,0x5AF11A03,0x2647A909,0x24
791387,0x5CFB4802,0x88CE4D29,0x353F5F5E,0x7038F851,0xF1F1C0AF,0x78EC6335,0xF2201A
D1,0xDF403561,0x4462DFC7,0xE22C5044,0x9C829EA3,0x43FD6EAE,0x7A42B3A7,0x5BFAAAE
C,0x3E046853,0x5789D266,0x3069CE1A,0xF115D008,0x4553AA9F,0x3194BE09,0xB4A9367D,0x
0A9DFEEC,0x7CA002D6,0x8E53A875,0x965E8183,0xE1219370,0x1FA480CF,0xD3FB6FEF,0xED
336CCB,0x9EE3CA39,0x9F224202,0x2D12D6E8,0xFAAC50CE,0xFA1E98AE,0x61498532,0x0367
8CC0,0x9E85EFD7,0x14D79DAC,0x0192B555,0x393BCE6B,0x232BA00D,0x84E18ADA,0x84557
BA7,0x56828948,0x166908F3,0x414A3437,0x7BB44897,0x2315BE89,0x7A01F224,0x7056AA5D,0
x121A3917,E3F47FA2,0x1F99D0AD,0x9BAD518B];

S2=[0xEB6E3836,0x9ED8A201,0xB49B5122,0xB1199638,0xA0A4AF2B,0x15F50A42,0x775F3759,
0x41291099,0xB6131D94,0x9A563075,0x224D1EB1,0x12BB0FA2,0xFF9BFC8C,0x58237F23,0x98
EF2A15,0xD6BCCF8A,0xB340DC66,0x0D7743F0,0x13372812,0x6279F82B,0x4E45E519,0x98B4B
E06,0x71375BAE,0x2173ED47,0x14148267,0xB7AB85B5,0xA875E314,0x1372F18D,0xFD105270,
0xB83F161F,0x5C175260,0x44FFD49F,0xD428C4F6,0x2C2002FC,0xF2797BAF,0xA3B20A4E,0xB
9BF1A89,0xE4ABA5E2,0xC912C58D,0x96516F9A,0x51561E77,,0x84DA1362,0x7A0E984B,0xBE
D853E6,0xD05D610B,0x9CAC6A28,0x1682ACDF,0x889F605F,0x9EE2FEBA,0xDB556C92,0x868
18021,0x3CC5BEA1,0x75A934C6,0x95574478,0x31A92B9B,0xBFE3E92B,0xB28067AE,0xD862D
848,0x0732A22D,0x840EF879,0x79FFA920,0x0124C8BB,0x26C75B69,0xC3DAAAC5,0xF1C871A
D,0x6C678B4D,0x46617752,0xAAE49354,0xCABE8156,0x6D0AC54C,0x680CA74C,0x5CD82B3F,
0xA1C72A59,0x336EFB54,0xD3B1A748,0xF4EB40D5,0x0ADB36CF,0x59FA1CE0,0x2C694FF9,0
x5CE2F81A,0x469B9E34,0xCE74A493,0x08B55111,0xEDED517C,0x1695D6FE,0xE37C7EC7,0x57
827B93,0x0E02A748,0x6E4A9C0F,0x4D840764,0x9DFFC45C,0x891D29D7,0xF9AD0D52,0x3F663
F69,0xD00A91B9,0x615E2398,0xEDBBC423,0x09397968,0xE42D6B68,0x24C7EFB1,0x384D472C,
0x3F0CE39F,0xD02E9787,0xC326F415,0x9E135320,0x150CB9E2,0xED94AFC7,0x236EAB0F,0x5
96807A0,0x0BD61C36,0xA29E8F57,0x0D8099A5,0x520200EA,0xD11FF96C,0x5FF47467,0x575C0
B39,0x0FC89690,0xB1FBACE8,0x7A957D16,0xB54D9F76,0x21DC77FB,0x6DE85CF5,0xBFE7AE
E9,0xC49571A9,0x7F1DE4DA,0x29E03484,0x786BA455,0xC26E2109,0x4A0215F4,0x44BFF99C,0
x711A2414,0xFDE9CDD0,0xDCE15B77,0x66D37887,0xF006CB92,0x27429119,0xF37B9784,0x9B
E182D9,0xF21B8C34,0x732CAD2D,0xAF8A6A60,0x33A5D3AF,0x633E2688,0x5EAB5FD1,0x23E
6017A,0xAC27A7CF,0xF0FC5A0E,0xCC857A5D,0x20FB7B56,0x3241F4CD,0xE132B8F7,0x4BB3
7056,0xDA1D5F94,0x76E08321,0xE1936A9C,0x876C99C3,0x2B8A5877,0x708784D6,0x13EB675F
,0x57592B96,0x07836744,0x3E721D90,0x26DAA84F,0x253A4E4D,0xE4FA37D5,0x9C0830E4,0xD
7F20466,0xD41745BD,0x1275129B,0x33D0F724,0xE234C68A,0x4CA1F260,0x2BB0B2B6,0xBD54
3A87,0x4ABD3789,0x87A84A81,0x948104EB,0xA9AAC3EA,0xBAC5B4FE,0xD4479EB6,0xC4108
568,0xE144693B,0x5760C117,0x48A9A1A6,0xA987B887,0xDF7C74E0,0xBC0682D7,0xEDB7705
D,0x57BFFEAA,0x8A0BD4F1,0x1A98D448,0xEA4615C9,0x99E0CBD6,0x780E39A3,0xADBCD40
6,0xA94BC384,0xF7A81CAE,0xAB84ECD4,0x00DEF340,0x8E2329B8,0x23AF3A22,0x23C241FA,
0xAED8729E,0x2E59357F,0xC3ED78AB,0x687724BB,0x7663886F,0x1669AA35,0x5966EAC1,0xD
574C543,0xDBC3F2FF,0x4DD44303,0xCD4F8D01,0x0CBF1D6F,0xA8169D59,0x87841E00,0x3C5
15AD4,0x6E71F2E9,0x9FD4AFA6,0x474D0702,0x8B6AD73E,0xF5714E20,0xE608A352,0x2BF644
F8,0x4DF9A8BC,0xB71EAD7E,0x6335F5FB,0x0A271CE3,0xD2B552BB,0x3834A0C3,0x341C590
8,0x0674A87B,0x8C87C0F1,0xFF0842FC,0x48C46BDB,0x30826DF8,0x8B82CE8E,0x0235C905,0x
DE4844C3,0x296DF078,0xEFAA6FEA,0x6CB98D67,0x6E959632,0xD5D3732F,0x68D95F19,0x43
FC0148,0xF808C7B1,0xD45DBD5D,0x5DD1B83B,0x8BA824FD,0xC0449E98,0xB743CC56,0x41F
ADDAC,0x141E9B1C,0x8B937233,0x9B59DCA7];

تصميم مولد مفاتيح انسيابي بواسطة خوارزمية اليراعة

محمد صالح مهدي
جامعة تكنولوجيا المعلومات والاتصالات
كلية معلومات الاعمال

نداء فليح حسن
جامعة التكنولوجية
كلية علوم الحاسوب

المستخلص :

التشفير الانسيابي هو احدى طرق التشفير للبيانات المرسلة عبر الانترنت. التشفير الانسيابي ملائم في الاتصالات وتطبيقات الوقت الفعلي. ان قوة قياس التشفير الانسيابي يعتمد على مدى عشوائية مولد المفاتيح المستخدم. اذا كانت عشوائية المفتاح الناتجة من مولد المفاتيح ضعيفة ، فان مفاتيح التشفير الانسيابي ممكن قراءتها والبيانات المشفرة بواسطة التشفير الانسيابي تكون غير محصنه للمهاجمين. هذا البحث يقترح اعتماد خوارزمية اليراعة (ذُبَاب يَطِير بِاللَّيْل يُضِيء ذَنَبُهُ) ودالة مفاتيح محلية لتوليد المفاتيح. المفاتيح المتولدة تكون مستقلة من النصوص الاصلية. ان نتيجة سلسلة المفاتيح من اليراع نجحت في الاختبارات القياسية الخمسة. ان مولد المفاتيح المقترح هو مبني على مفهوم الكتلة حيث يكون أفضل استخدامه للتطبيقات الوقت الفعلي بدلا من بناءة على مفهوم البث. من ناحية أخرى، ان مولد المفاتيح المقترح يحقق المتطلبات الثلاثية القياسية من ارتباط عالي، عشوائية قوية وتعقيد عالي.

الكلمات المفتاحية: التشفير الانسيابي، مولد مفاتيح، خوارزمية اليراعة، دالة مفاتيح محلي .