Volume 3, Issue 6, June 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Public Key Cryptography Using Particle Swarm Optimization and Genetic Algorithms

**Smita Jhajharia**
*Visiting Faculty, Computer Engineering*
*Delhi Technological University, DTU*
*Delhi, India*

**Swati Mishra**
*Student, Computer Engineering*
*Delhi Technological University, DTU*
*Delhi, India*

**Siddharth Bali**
*Student, Computer Engineering*
*Delhi Technological University, DTU*
*Delhi, India*

***Abstract— This paper proposes an algorithm for Public Key Cryptography (PKC) using the hybrid concept of two evolutionary algorithms, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) respectively. PSO alone are fast and easy to implement, they follow the procedures of common evolutionary algorithm and posses memory feature which is absent in GA making it more valuable. In GA whole population or set of individual chromosome work together sharing information to reach an optimal solution whereas PSO focuses on only the best possible solution. Particles in PSO converges in small optimal area quickly. The random variables needed for PSO initialization are provided by GA ensuring that every time algorithm runs a new unique random value is initialized which cannot be guessed. PSO uses a set of fine fit initial keys as input from key domain generated by GA and outputs the position of key having the highest fitness among the keys. Thus, PSO-GA algorithm aims here for generating the fittest among the fine fit keys in key domain containing best keys of highest possible strength. The results produced by this hybrid algorithm to be tested for frequency test, gap test, auto-correlation test, binary derivative test, change point test, serial test, run test and also to check for the linear complexity of key proving its validity and practical use of the proposed work in PKC.***

***Keywords— Public-Key Cryptography (PKC), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Fitness Function, Key Space, Genetic Crossover, Genetic Mutations, Evolutionary Algorithm, Artificial Intelligence, Optimisation, Swarm Intelligence, Key Generation***

## I. INTRODUCTION

In today's era of Information & Technology, communication through Internet involves electronic transactions, mails etc. Sending & receiving information to others thus demands electronic security. Information transmitted includes critical as well as personal data which every user desires to be safe and secure from illegal interception. Cryptography is the field of transforming messages i.e. plaintext into unreadable format i.e. ciphertext using encryption which can be converted back to original messages using decryption by the concerned recipient. Cryptography is of two types : Symmetric Key Cryptography which uses single key for both encryption & decryption, and Asymmetric Key Cryptography which use two keys, one used for encryption i.e. public key and another used for decryption i.e. private key. The transmitted message must entertain the basic requirement of Authentication : confirming identity of sender, Confidentiality : unauthorized disclosure of contents, Data Integrity : Ensuring no modification to intended message, Authorization, and Non-Repudiation : for proof of transmission and delivery. Particle Swarm Optimization (PSO) is sturdy optimization technique established on intelligence of swarm and its movement. PSO adopts the policy of social interaction and natural systems like bird flocking, fish schooling, ant colony etc. for solving problem. PSO technique was originally given by Eberhart and Kennedy. It's amongst the latest techniques encountered in evolutionary optimizations. Potential solutions of PSO algorithm are referred to as particles which move about in multi-dimensional search space. During movement each particle's position are adjusted according to its previous best position and the globally best position. It's a solitary evolutionary algorithm technique with absence of fittest based survival. PSO is clean yet compelling search technique and one of the successful example of artificial intelligence and swarm intelligence systems which has solved many different problems including search & optimization problems. Some of the application include in engineering design, game theory and hydrologic problems etc. This paper presents another novel application as PKC cryptography.

Genetic Algorithms (GA) is a class of probabilistic optimization technique with good heuristics that is usually applied to discrete optimization problems. GAs are inspired by the natural evolution process. GA can deliver fast and favourable solutions over a large search space. It is used for solving various problems which cannot be solved by applying other routines. GA works on population of individuals called chromosomes which represent feasible solution to a given problem. It has the ability to run on parallel machines and with adequate designs the population tends to converge for an optimal solution. GA evolve by iteratively applying a set of stochastic operations on chromosomes. During each iteration, it starts by performing crossover i.e. combining good properties from two parents. Then mutation is performed on child chromosomes followed by calculation of their fitness. Every individual from population is symbolized by fitness function. More surpassing is the fitness the better is the solution. Depending on fitness parents are selected from population to reproduce a new generation. Individuals with high fitness are more fortunate to reproduce. This paper proposes and implements a novel approach of PKC using the assets of both PSO and GA. Our algorithm depicts how with the aid of GA to Swarm, we can obtain key domain for PKC that possess pure randomness assisted by improved fitness and exhibiting better statistical test results for the keys generated for PKC. Thus generating quick yet effective keys for PKC having practical implementation.

## II. Literature Review

This work makes the proposition of Public Key Cryptography (PKC) with the combined assistance of Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Literature Review has been done and information regarding work and progress in the context of our work has been observed. It was found that PKC using swarm as solitary has been carried for wireless communication [1] in which tests were performed using deterministic approach. It was observed that encryption based on chaotic map has been derived from a simplified model of Swarm Intelligence (SI) [2]. In one of the work SI based key generation has been performed for encryption of data in cellular network [3] using Ant Colony Key Generation Algorithm (AKGA), SI approach. Similarly, in another paper encryption of plaintext using a stream cipher method [4] was adopted using AKGA again with sole SI. Swarm based cryptanalysis called Ant-crypto for Data Encryption Standard (DES) has been displayed by applying Binary Ant Colony Optimization (BACO) [5]. Cryptanalysis has also been focused for S-DES using PSO along with GA parameters. It is to be noted that not much attempt has been made for enhancing the quality of keys produced by PSO along with the aid of GA to generate better keys possessing pure randomness with no relation to prior keys obtained and passing the statistical tests to establish its practical application.

## III. Algorithm For Public Key Cryptography Using Particle Swarm Optimization And Genetic Algorithms

The high key strength is marked by mutation and crossover operation of GA and also by mechanism of PSO to prove together that PSO and GA acts as a strong unit compared to other EA. Initial particles i.e. keys generation for PSO is supported by exploiting GA technique producing 192-bits initial keys which are unique and non-repeating forming a very strong unique key domain space. PSO is employed to select the best keys among the contender key domain produced by GA.

### *A. GENERATION OF KEYS USING GENETIC ALGORITHM*

A domain of keys is obtained by running the following algorithm of key generation using GA as proposed :

*Step 1. Initialization :* GA is started with an initial population of 192 bits chromosomes which are generated randomly. This population is stored in an array, initPop[MAX_POPULATION][192] such that each cell consists of binary values 0 or 1. MAX_POPULATION of such chromosomes is taken to be 200 here. The size of chromosome cell is equivalent to the length of key intended to be generated.

*Step 2. Threshold Check :* Chromosomes obtained from Step 1 should be within a standard threshold value. Thus, for each chromosome individual a number is assigned which is monitored against the threshold value. If the chromosome excels this threshold value, it is considered suitable for subsequent steps.

*Step 3. Iterative application of stochastic operators to obtain final generation :* Generation of chromosomes individual is achieved by combined efforts of stochastic operators i.e. selection, crossover and mutation followed by the calculation of fitness. After each such iteration end result is saved in an array finalPop[MAX_POPULATION][192].

Following steps are carried sequentially till the size of array, final population[][] meets MAX_POPULATION limit :

*1) Parent Selection :*

Two parents in the search space are obtained from the initial population created in step 1. Selection of both parents is random in nature. Chromosomes in upcoming generation, i.e. child chromosomes relies on parents selected here.

*2) Crossover Operator :*

Parents are crossed by employing one-point ring technique for crossover. A random cut-point is generated in this ring and children are obtained by reading the ring clockwise for first child and anti-clockwise for second child from the cut-point. Threshold check as done in step 2 is carried.

*3) Mutation :*

Taking the account of probability of child chromosome randomness from the expected trend mutation operator is used to mimic such natural behaviour. Bits undergoing mutation in chromosome is obtained using a random function over the key length. Total mutations to be performed in a chromosome is obtained by solving (1). Again threshold check is done as in step 2.

$$\text{Number of Mutation} = \frac{(\text{No. of cells in a chromosome} * \text{No. of chromosomes} * \text{mutation rate})}{200} \qquad (1)$$

*Step 4. Key Fitness :* Fitness value is calculated for each chromosome i.e. 192 bit key present in final population using the following routine [7] :

*1) Shannon Entropy Calculation :*

$H(X) = -\{ p * (\log_2(p)) \} - \{ (1-p) * (\log_2(1-p)) \}$ (2)

where p refers to the percentage of randomness of final population over the initial population.

*2) Chi Square test of Shannon Entropy :*

Chi-square value, $X^2 = \dfrac{(\text{observed\_H} - \text{expected\_H})^2}{(\text{expected\_H})}$ (3)

where expected_H = 0.5 considering when bits are purely random alternate 0's and 1's.

*3) Phi-Coefficient Calculation :*

Phi-Coefficient is calculated for each chromosome in final population array. Phi-coefficient also called Coefficient of Auto-correlation (4) plays a vital role in selection of final keys domain.

Phi-Coefficient, $\phi^2 = X^2 \div \text{MAX\_POPULATION}$ (4)

*Step 5. Determination of best fit key :* Chromosomes present in final population are ranked according to their fitness i.e. Phi-Coefficients. The chromosome with highest rank is considered to be the best fit 192-bit key.

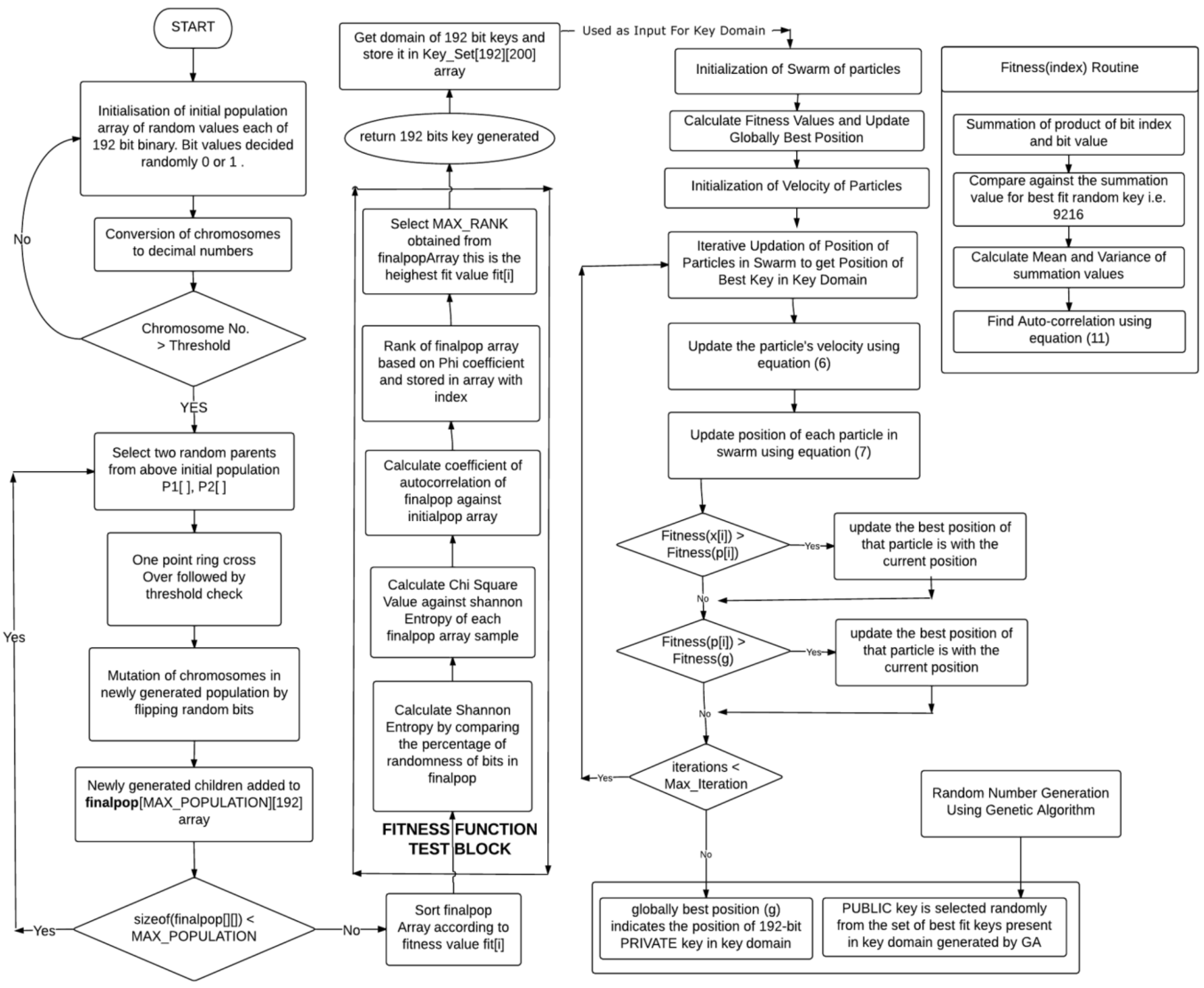
START
Initialisation of initial population array of random values each of 192 bit binary. Bit values decided randomly 0 or 1 .
Conversion of chromosomes to decimal numbers
Chromosome No. > Threshold
No
YES
Select two random parents from above initial population P1[ ], P2[ ]
One point ring cross Over followed by threshold check
Mutation of chromosomes in newly generated population by flipping random bits
Newly generated children added to finalpop[MAX_POPULATION][192] array
sizeof(finalpop[][]) < MAX_POPULATION
Yes
No
Sort finalpop Array according to fitness value fit[i]
Calculate Shannon Entropy by comparing the percentage of randomness of bits in finalpop
Calculate Chi Square Value against shannon Entropy of each finalpop array sample
Calculate coefficient of autocorrelation of finalpop against initialpop array
Rank of finalpop array based on Phi coefficient and stored in array with index
Select MAX_RANK obtained from finalpopArray this is the heighest fit value fit[i]
FITNESS FUNCTION TEST BLOCK
return 192 bits key generated
Get domain of 192 bit keys and store it in Key_Set[192][200] array
Used as Input For Key Domain
Initialization of Swarm of particles
Calculate Fitness Values and Update Globally Best Position
Initialization of Velocity of Particles
Iterative Updation of Position of Particles in Swarm to get Position of Best Key in Key Domain
Update the particle's velocity using equation (6)
Update position of each particle in swarm using equation (7)
Fitness(x[i]) > Fitness(p[i])
Yes
update the best position of that particle is with the current position
No
Fitness(p[i]) > Fitness(g)
Yes
update the best position of that particle is with the current position
No
iterations < Max_Iteration
Yes
No
globally best position (g) indicates the position of 192-bit PRIVATE key in key domain
PUBLIC key is selected randomly from the set of best fit keys present in key domain generated by GA
Random Number Generation Using Genetic Algorithm
Fitness(index) Routine
Summation of product of bit index and bit value
Compare against the summation value for best fit random key i.e. 9216
Calculate Mean and Variance of summation values
Find Auto-correlation using equation (11)

Fig. 1 Flow chart of Public Key Cryptography using PSO and GA.

## *B. PUBLIC KEY CRYPTOGRAPHY USING PSO and GA*

*Input :* Algorithm parameters (Swarm_Size, Max_Iter, c1, c2, w, V_MAX), Domain of Keys (Key_Set[][]) initialized with keys by running GA, Random Numbers generated from GA (r1, r2).

*Output :* The key having the highest fitness as found by PSO.

*Algorithm Steps :*

*Step 1. Initialization of Swarm of particles :* Each particle in swarm is characterized by its position, velocity and best position. Initial position and best position of particles in swarm is assigned random value between 0 to size of key domain i.e. 200 here. Initially, Globally best position is assumed to be the position of first particle in the swarm.

*Step 2. Update Globally Best Position :* Globally best position is updated with the best position of ith particle in swarm if fitness value of key present at best position of ith particle is found to be more than the fitness value of key present at globally best position in Key_Set[][] array.

*Step 3. Initialization of Velocity of Particles :* Each particle in swarm is initialised with a random seed velocity which is bounded by the intervals V_MAX and V_MIN. Upper and lower bounds on velocity of each particle in swarm are determined by the size of key domain i.e. Key_Set[][].

*Step 4. Iterative Updation of Position of Particles in Swarm to get Position of Best Key in Key Domain :* Each particle in swarm learns from its own previous history and the best position of its neighbour particles. Following steps are repeated until best key's position is found :

1) *Update the particle's velocity :*

v[i] = w * v[i] + c1 * r1 * (p[i] - x[i]) + c2 * r2 * (g - x[i]) (5)

where, v[i] is velocity of ith particle, x[i] is position of ith particle, p[i] is ith particle's best position, g is globally best position, r1 and r2 are initialized with random numbers produced using GA, w is inertia weight = 1.2, c1 is self-confidence and c2 is swarm confidence. c1 and c2 are initialised between 1.5 - 2.0

2) *Update position of each particle in swarm :*

x[i] = x[i] + v[i] (6)

where x[i] is position of ith particle and v[i] is velocity of ith particle.

3) *Update best position of particles :*

If new position of particle x[] represents a key with higher fitness in key domain than the fitness of key represented by its previous best position p[], then the best position of that particle is updated with the current position i.e. if ( Fitness(x[i]) > Fitness(p[i]) ).

4) *Update globally best position :*

Globally best position is also updated if fitness of the key present at previous globally best position is less than the fitness of the key present at best position of particle under consideration i.e. if ( Fitness(p[i]) > Fitness(g) ).

*Fitness Calculation :* Fitness function calculates the fitness of key stored at a particular location in key domain. Best randomness is represented by alternative sequence of 0s and 1s. Summation of product of bit index and bit value is calculated and compared against the summation value for best fit random key i.e. 9216. If there is difference in summation values then auto-correlation (10) is calculated which is returned as the fitness value.

diff1 = abs( summation1 - mean(summation1,summation2) ) (7)

diff2 = abs(summation2 - mean(summation1,summation2) ) (8)

$$\text{variance} = \frac{((\text{diff1})^2 + (\text{diff2})^2)}{2} \quad (9)$$

$$\text{FF} = \frac{\text{diff1*diff2}}{(\text{variance*variance})} \quad (10)$$

where abs() represents absolute function, mean(a,b) represent function to calculate mean of a & b and FF represents the fitness value.

*Step 5.* After the final iteration, globally best position (denoted by g) indicates the position of 192-bit PRIVATE key in key domain which is of best quality. PUBLIC key is selected randomly from the set of best fit keys present in key domain generated by GA.

## IV. Results And Analysis

The proposed work's implementation demonstrated to show how the PKC using PSO and GA algorithm developed is a suitable match for generation of keys of higher fitness for quality cryptography. The keys are found to be non-computable easily, large keys of variable 192 bits having no relation to prior history of keys been generated during iterations. Rigorous testing were carried for keys generated intended to be designated for PKC by performing standard statistical tests. The work has been implemented in C platform and statistical tests were carried on the key generated by each iteration of the algorithm to ensure and authenticate that key passed various tests and proposed idea of PSO for the best fit keystream among the key domain specified by GA aptly.

### A. Binary Derivative Test

Binary derivative test was performed to analyze 192-bit keys for randomness of binary bits.

TABLE I
BINARY DERIVATIVE TEST

| **Sample Number** | **best_key_index** | $p(best\_key\_index) = \frac{Counter}{(192.0 - k)}$ |
|---|---|---|
| 1 | 79 | 0.4995390625 |
| 2 | 159 | 0.491457802083334 |
| 3 | 31 | 0.506282010416667 |
| 4 | 102 | 0.509768895833333 |
| 5 | 60 | 0.502609515625 |
| 6 | 122 | 0.502665291666667 |
| 7 | 180 | 0.50586365625 |
| **Average Value of p** | | 0.502598033482143 |

The key bits were XORed with each pair of consecutive bits as operands until only a single bit was left. Table I sights the key under test has appearance of randomness as average value of 0.5 which is perfect in consideration with alternate key bit sequence of 1's and 0's thus each having equal probability of 0.5.

### *B. Change Point Test*

Table II shows actual significance probability of the change in the proportion of ones in 192 bit keystream which is found to be less than 1.00 thus confirming the change point test. The change point was checked i.e. the maximum difference between the proportions of ones including and after the point.

TABLE II
CHANGE POINT TEST

| | | | | | | | **Average** |
|---|---|---|---|---|---|---|---|
| **Total Bits = n** | 192 | 192 | 192 | 192 | 192 | 192 | **192** |
| **Total No. of 1s** | 95 | 98 | 99 | 96 | 97 | 104 | **98.1667** |
| **Change Point (t)** | 152 | 28 | 6 | 60 | 18 | 12 | **46** |
| **Total No. of 1s before t** | 69 | 12 | 3 | 30 | 7 | 8 | **21.5** |
| **Proportion of 1s before t** | 0.4539 | 0.4285 | 0.5 | 0.5 | 0.3888 | 0.6666 | **0.4896** |

### *C. Serial Test*

Serial test also known as equidistribution test was performed on samples of key generated.

TABLE III
SERIAL TEST

| | **Observed (x1)** | **Expected (x2) = 192/4** | **Difference = (x1–x2)** |
|---|---|---|---|
| **n00** | 52 | 48 | 4 |
| **n01** | 44 | 48 | -4 |
| **n10** | 43 | 48 | -5 |
| **n11** | 52 | 48 | 4 |
| **Sum of Difference (D_SUM)** | | | -1 |
| **Error_Serial = abs(D_SUM)** | | | 1 |

Table III depicts that the observed and expected values of the occurrence of successive 00, 01, 10, 11 are found to be very close with approximately zero error percentage thus showing that the consecutive bits in keys are independent of each other.

### *D. Run Test*

True randomness of key stream produced was tested by run test.

TABLE IV
RUN TEST

| **Best Key Index** | **No. of runs of 1s = p** | $Z = \frac{(Expected\ no.of\ runs - Observed\ no.of\ runs)}{\overline{X}(mean\ no.of\ runs)} = \frac{(96-p)}{48}$ | **Summation** |
|---|---|---|---|
| 101 | 53 | 0.447917 | 9211 |
| 180 | 58 | 0.395833 | 9224 |
| 42 | 50 | 0.479167 | 9212 |
| 34 | 50 | 0.479167 | 9247 |
| 33 | 52 | 0.458333 | 9152 |
| 52 | 53 | 0.447917 | 9218 |
| 173 | 50 | 0.958333 | 9159 |
| 140 | 53 | 0.895833 | 9204 |
| **Maximum Z** | | **0.958333** | |
| **Z critical (for alpha)** | | **0.05** | |
| **Z (critical)** | | **1.96** | |

The calculated run test value z was found less than the critical value of 1.9 as shown in table IV. Hence results pass run test and it showed the pure random distribution of randomly generated key stream.

### *E. Gap Test*

It can be seen in Table V, that the value of best key index and summation values for the keys generated are different. Hence, non-repeating behavior of keys was observed on analysis using gap test.

TABLE V
GAP TEST

| Sample No. | Best Key Index | Summation (Key Value) |
|---|---|---|
| 1 | 97 | 9280 |
| 2 | 98 | 9115 |
| 3 | 150 | 9224 |
| 4 | 40 | 9207 |
| 5 | 122 | 9204 |
| 6 | 131 | 9185 |
| 7 | 25 | 9216 |
| 8 | 160 | 9214 |
| 9 | 44 | 9216 |
| 10 | 36 | 9172 |

*F. Frequency Test*

To check the anomaly of expected key stream and obtained key stream, frequency test also known as chi-square test was performed.

TABLE VI
FREQUENCY TEST

| Serial No. | Observed (d1) | Expected (d2) | $P = \frac{(d1-d2)}{(d1+d2)}$ | Observed H(X) | Expected H(X) | $X^2$ | $\varphi = \sqrt{\frac{X^2}{n}}$ |
|---|---|---|---|---|---|---|---|
| 1 | 8 | 107 | 0.86086956521 | 0.5819569 | 0.5 | 0.0134338867 | 0.051834133129 |
| 2 | 96 | 119 | 0.10697674418 | 0.4907290 | 0.5 | 0.0001719024 | 0.005863487783 |
| 3 | 36 | 37 | 0.01369863013 | 0.1044190 | 0.5 | 0.3129685316 | 0.250187342482 |
| 4 | 96 | 77 | 0.10982658959 | 0.4993926 | 0.5 | 0.0000007376 | 0.000384103922 |
| 5 | 90 | 17 | 0.68224299065 | 0.9019255 | 0.5 | 0.3230883600 | 0.254200062945 |
| $Average\ X^2$ | | | | | | **0.1299326837** | |

Standard chi square table was referred for expected frequency for value of alpha 0.01 against randomly generated key using the algorithm. Table VI shows here the obtained results were within acceptable limit with average chi square value i.e. $X^2$ to be less than 6.663.

*G. Linear complexity of key stream*

Stream of bits of 0s and 1s in key samples were analyzed for linear complexity.

TABLE VII
LINEAR COMPLEXITY OF KEY STREAM

| Sample No. | Linear Span |
|---|---|
| 1 | 96 |
| 2 | 96 |
| 3 | 95 |
| 4 | 97 |
| 5 | 96 |
| 6 | 97 |
| **Observed Average** | **96.167** |
| **Expected Average** | **96** |

As shown in Table VII, observed linear span was found to be approximately equal to the expected linear span. Hence, unpredictable sequence of 0s and 1s are generated such that linear complexity profile comes out to be close to the $\frac{n}{2}$ -line i.e. $\frac{n}{2} = \frac{192}{2} = 96$ where n denotes number of bits in a key sample.

## V. CONCLUSIONS

PSO using GA was proved to be favorable in selection of the best possible key from the key domain. All the statistical tests results were passed proving the effectiveness and efficiency of the combined use of PSO and GA than other methods used for finding global optimum. The final keys obtained were unique, random, non-repeating and cryptographically strong in nature with large linear complexity and high order of auto-correlation. Thus, keys produced are also safe from related key attacks and weak key attacks. Keys successfully passed frequency test, gap test, auto-correlation test, binary derivative test, change point test, serial test, run test and the linear complexity test. The proposed approach showed the encouraging collaboration of two effective evolutionary algorithms PSO and GA thus forming a strong algorithm. To ensure best results are obtained long iterations of algorithm were implemented further increasing the complexity of keys generated. The fast nature of algorithm considerably reduces encryption time even on large stream of plaintext. Better test results proves the generation of more random keys. It was also observed that no such plaintext can be chosen which shall reveal the key in cipher text because keys are purely random and independent of plaintext. Brute Force Attack on 192-bit keys would require $2^{192}$ different combinations to be checked for guessing the key. Keys generated can also be used as input to hash function for MAC and generation of Message digest. The algorithm enjoys the add-on benefits of GA which were lacked by PSO. A number of issues related to PSO are solved in our algorithm by combining the effectiveness of GA. Vigorous settings are done for various parameters of PSO using GA random number generator. GA is also used in generation of key domain of best fit keys passing auto-correlation fitness test. Simple but powerful self-improved algorithm obtained having its practical implementation solving optimization problems.

TABLE VIII
ALGORITHM PARAMETERS

| **PSO Parameters** | | **GA ( Key Domain Generator) Parameters** | |
|---|---|---|---|
| **Total No. of Keys** | 200 | **Population Size** | 200 |
| **KEY_SIZE** | 192-bits | **Crossover Type** | One-Point Ring Crossover |
| **Inertia Weight (w)** | 0.5 | **Mutation Rate** | 0.5 |
| **Cognitive Weight (c1)** | 2 | **No. of mutation** | $\frac{(No.of\ cells\ in\ a\ chromosome * No.of\ chromosome * mutation\ rate)}{200} = \left(\frac{192*200*0.5}{200}\right) = 96$ |
| **Social Weight (c2)** | 2 | | |
| **r1, r2** | GA Random No. Generator | | |
| **No. of particles in the swarm, SWARM_SIZE** | 10 | | |
| **Maximum No. of iterations, Max_iter** | 100 | | |

Experimental results demonstrates that with the combined effectiveness of PSO & GA, optimum keys can be produced within less time. If algorithm is run on high speed computers, it will further reduce the total time consumption to produce high quality keys. Hence, the proposed algorithm can be parallelized for higher performance by implementing on GP-GPU and cluster computers.

## ACKNOWLEDGMENT

The authors thank and gratefully acknowledges the support of Miss. S. Jhajharia, Research Scholar DRDO, Visiting Faculty DTU, Delhi.

## REFERENCES

[1] Arindam Sarkar, J. K. Mandal "Swarm Intelligence based Faster Public-Key Cryptography in Wireless Communication (SIFPKC)". International Journal of Computer Science & Engineering Technology (IJCSET).

[2] Hussein, R.M.; Ahmed, H.S.; El-Wahed, W., "New encryption schema based on swarm intelligence chaotic map," Informatics and Systems (INFOS), 2010 The 7th International Conference on , vol., no., pp.1,7, 28-30 March 2010.

[3] Sreelaja, N.K.; Pai, G.A.V., "Swarm intelligence based key generation for text encryption in cellular networks," Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on , vol., no., pp.622,629, 6-10 Jan. 2008 doi: 10.1109/COMSWA.2008.4554485.

[4] Sreelaja, N. K. and Vijayalakshmi Pai, G. A. (2011), Swarm intelligence based key generation for stream cipher. Security Comm. Networks, 4: 181–194. doi: 10.1002/sec.132.

[5] Salabat Khan, Armughan Ali and Mehr Yahya Durrani, “Ant-Crypto, a Cryptographer for Data Encryption Standard”. IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 1, January 2013 ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814.

[6] Vimalathithan R., M. L. Valarmathi, “Cryptanalysis of Simplified-DES using Computational Intelligence”, WSEAS TRANSACTIONS on COMPUTERS.

[7] S. Mishra, S. Bali “Public Key Cryptography Using Genetic Algorithm”. International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-2, Issue-2, May 2013.

[8] *Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks IV. pp. 1942–1948. doi:10.1109/ICNN.1995.488968.*

[9] Shi, Y.; Eberhart, R.C. (1998). "A modified particle swarm optimizer". Proceedings of IEEE International Conference on Evolutionary Computation. pp. 69–73.

[10] Kennedy, J. (1997). "The particle swarm: social adaptation of knowledge". Proceedings of IEEE International Conference on Evolutionary Computation. pp. 303–308.

[11] Kennedy, J.; Eberhart, R.C. (2001). Swarm Intelligence. Morgan Kaufmann. ISBN 1-55860-595-9.

[12] Goldberg, D E., Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA. : Addison-Wesley, 1989. Stallings, W., “Cryptography and Network Security : Principles and Practice”, 3rd Edition. Prentice Hall. Boston Columbus Indianapolis.

[13] Bruce Schneier, Applied Cryptography, 2nd edition, Wiley, 1996, ISBN 0-471-11709-9.