International Journal of Information & Network Security (IJINS)
Vol.2, No.1, February 2013, pp. 17~31
ISSN: 2089-3299

Evolutionary Computation Guided Energy Efficient Key Organization in Wireless Communication (ECEEKO)

Arindam Sarkar*, J. K. Mandal*
* Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, Pin- 741235, W.B, India.

Article Info

Article history:

Received Aug 2th, 2012
Revised Sept, 7th, 2012
Accepted Oct, 22th, 2012

Keyword:

Evolutionary Computation
Wireless Communication
Key Organization
Energy Efficient

ABSTRACT

In this paper an Evolutionary Computation based energy efficient novel key organization (ECEEKO) policy has been proposed by incorporating computational safety, power management and restricted usage of memory in wireless communication. Generating functions of secured key are generated from the sink node through evolutionary computation and distribute to the header and sensor nodes. On agreeable particular power consumption constraints each potential key generating function (KGF) is encoded as a chromosome. Using entropy measure fitness is calculated to assess key distribution. Common keys are assembling by headers and sensor nodes. Performance analysis of the ECEEKO approach and comparison study with several existing methods has been done.

Corresponding Author:

Arindam Sarkar,
Department of Computer Science and Engineering,
University of Kalyani, Kalyani, Nadia, Pin- 741235, W.B, India.
Email: arindam.vb@gmail.com

1. INTRODUCTION

Hierarchical sensor networks (HSN) [4] is a category of a wireless sensor networks (WSN) [2, 5, 10, 18] comprises of sensor nodes, header and sink nodes. Another category of WSN is distributed sensor networks (DSN) which is consist of sensor nodes and sink node. In HSN header and sink node are differ in terms of energy, memory and computational capability. In sensor nodes encryption algorithms are infeasible because sensor nodes are limited by computing capability, memory size and battery life. So, sensor nodes cannot handle complex exponentiation for encryption using limited battery life. Security of WSN is essential when sensor nodes are deployed in hostile and malicious environment. But quite a few security schemes mainly suffer from optimal usage of memory space and energy. Proposed scheme handles this problem by optimizing memory usage and power consumption using evolutionary computation.

2. PROBLEM STATEMENT

Several security schemes have been proposed for pre distribution of key [6, 7, 8, 12, 13, 15, 21], group key distribution [11], integrated hierarchical key deployment [9] and key renewal [6, 14, 16] in sensor networks. But this security schemes suffers from efficient power and memory usage. To resolve this problem a novel and efficient technique has been proposed in this paper which optimizes the power and memory usage and provides good computational security using evolutionary computation.

3. RELATED WORK

A number of security algorithms were proposed for providing security in sensor network. Blom's proposed key pre-distribution method with key size k and network size N which takes k*N memory space to reach full connection that is quite large [21]. To reduce memory usage in Blom's scheme a new scheme has been proposed by DU et al. known as DDHV-D [3]. In this scheme the neighboring nodes are checked for storing key. A node did not waste its memory by storing key of a node which is not its neighbor. This scheme also uses deployment knowledge of reducing memory usage at the time pre distribution of key for constructing neighboring connection. Another scheme proposed by Chien-Lung Wang et al. [1] has also some shortcomings like if there are m number of genes in a gene pool then total number of possible chromosomes (KGF) is only m!, so total m! number of keys are possible from m! chromosomes and f (f<<m!) keys are selected for evaluation. Also this technique does not support mutation within a gene. Rui Zhou et al. [23] proposed a novel key management scheme, which utilizes Elliptic Curve Cryptography and the t-degree trivariate symmetric polynomial in the design of an efficient key management scheme for sensor nodes. Chen Chen et al. [24] have proposed a scheme which follows the architecture of the three layers in LOCK.

In this paper, evolutionary computation guided optimized power and efficient memory usage key organization scheme in a hierarchical sensor network has been proposed. In this method using evolutionary computation strategy key generating functions (KGF's) i.e. chromosomes are generated by sink nodes for rekeying on sensor nodes under energy consumption constraints. The functions are further divided into genes which are then embedded into sensor nodes and headers before deployment. As sensor nodes are deployed, the headers will randomly assemble the common genes and send the series to sensors for rebuilding the KGFs (chromosomes) for rekeying. The rekeying functions are rebuilt in each predefined interval, such that it would be difficult for an attacker to crack the functions in time. Besides, it is also energy-efficient in that it consumes the energy for rekeying only while assembling a new key in a sensor node. The proposed evolutionary computation rekeying scheme can thus not only meet the constraints of low energy and little memory but also fulfill the security requirement.

4. PROPOSED ECEEKO SCHEME

In ECEEKO technique shortcomings of the existing methods has been resolved by introducing following features:

a) In ECEEKO technique each gene is made of operands and operators. For example gene G_1= + p q. i.e. (p + q). Collection of such type of genes constructs a gene pool. For example Gene $pool_1$= {G_0, G_1, G_2,...., G_m}. Where gene G_0 consist of linking operators like +, -,* etc for concatenating at most m number of genes satisfying power consumption constraints. Initially this scheme consider a set of gene pools like {Gene $pool_1$, Gene $pool_2$, ... , Gene $pool_p$}. If a gene pool has m number of genes and n number of linking operators then this gene pool can produce (m! + n! -1) number of different chromosomes by linking each gene with other in different combination using linking operators. From this (m! + n! -1) number of different chromosomes only f (f << (m!+n!-1)) chromosomes are selected for evaluation because evaluation of distribution of this (m!+n!-1) chromosomes are impossible. Entropy measurement which is basically a fitness function is used for selecting best chromosome among other chromosomes in a particular gene pool. This best fitted chromosome is known as KGF and value produce from this KGF is considered as a key. In this way using evolutionatry computation from p number of different gene pools at most p number of best fitted KGF can be constructed. Again from this p number of KGF's best KGF having maximum entropy value is selected. Now genes belongs to the best fitted KGF's are transmitted to the header nodes. Header nodes can use different combination of these genes for producing different key value. Sensor nodes belonging to a common header node use same gene combination for producing a common key value which is used by all sensor nodes under this header node.
b) In chromosome construction strategy one new gene G_0 consisting of rules for concatenating genes has been introduced.
c) Concept of context has also been introduced in ECEEKO. Where contexts are formed by collecting genes having same type of input and returning same type of arguments. If genes G_1 and G_2 have operators that take real arguments and return Boolean values then these two genes will belongs to same context. Similarly, genes having operators which takes Boolean arguments and returning Boolean values are also belongs to same context.
d) In ECEEKO if there are m number of genes and n numbers of conacatinating operators then total possible chromosomes i.e. KGF's (different gene combination) will be (m! + n! - 1). Whereas Chien-Lung Wang et al. proposed method can generate only m! where m is the number of code slices(genes).
e) Crossover of muligenic chromosome with different contexts has been proposed in ECEEKO.

f) Mutation within a gene as well as within a chromosome has also incorporated in the proposed method. If mutation point falls within a gene then at the mutation point the exchanged symbols must come from the same context of operands and operators.

In this hierarchical sensor network model a hierarchical tree structure is maintained where sink node is placed at the root. Header nodes are become intermediate place holder of the tree and all the sensors nodes are placed at the leaf level.

- Sink node (Root node) - It acts as a base station and provides the access to the outside for the WSN. KGF's are generated from the sink for generating keys in the rekeying strategy. Sink node initially starts with number of genes, and then find out the good combination of genes (chromosome) using evolutionary computation. Set of binary operators and operands are use to form a gene. Several genes are concatenated in several combinations using predefine binary operators to form a KGF (chromosome). This sink node sends m genes belong to the best chromosome (KGF) to the header nodes before deployment. It is assumed that the sink nodes are secure and powerful. It means that the sink node is located in a secure place and has limitless computational capability, power, memory, and communication bandwidth.

- Header node (Intermediate node) – It has larger memory size, more amounts of energy and more powerful computational capability than sensor nodes. It delivered the aggregated results to sink node after combining the monitored data from the sensor nodes. The function of header nodes to recognize which set of sensor nodes belongs to their area and select common set of genes combination (chromosome) for the sensor nodes belonging to a particular header node and transfer them to the sensor nodes.

- Sensor nodes (Leaf node) - It has small amount of power and its functionality is to monitoring the environment and sends the gathered information to the header nodes. Each sensor node generates new key using several possible gene combination. The functions of sensor nodes include the execution of simple operations, like bitwise operations, fundamental arithmetic operations, and shift operations, and generate keys according to the instructions from headers. Initially it is assume that before the first rekeying a sensor node is completely secure. It means in within a certain *t* time, an adversary can not compromise any sensor node. An adversary may, however, compromise any sensor node after the *t* time. The sensor nodes and their headers thus have to finish the first rekeying process within *t* time.

In ECEEKO scheme each gene comprises of one binary operator and two operands. Different genes are concatenated using different operators and forms different chromosomes. Different chromosomes have different key distributions and energy consumptions. In this scheme evolutionary computation is used to find out good set of KGF having a key distribution as uniform as possible under low energy consumption. Distribution of key value is observed by proposed entropy calculation.

Proposed ECEEKO scheme has 2 phases. These are

- Construction of good KGF: In this phase an initial population is created using c numbers of chromosomes each having m numbers of genes. These m genes are concatenated with each other in different possible combination with the help of predefined set of binary operators to form pool of chromosomes. This pool comprises of all the combination of predefined operators and genes. Depend on power consumption constraints appropriate chromosomes get selected for construction of mating pool. Different evolutionary operators like crossover and mutation are applied on best parent chromosomes to form new child chromosomes. In this way good offspring gets created and used as KGF.
- Rekeying: In this phase best chromosomes generated from previous phase delivered to the headers and sensors for rekeying.

4.1 Chromosome Construction

Each set of genes is encoded as a chromosome. Several encoding techniques have already been proposed earlier [17, 20]. Genes are the predefined set of binary operators and operands generated by the sink nodes for constructing KGFs. In a simple scenario, to obtain the chromosome, sub-expressions encoded by the genes may be simply aggregated by a linking function. A more advanced version uses an extra special gene made up of operators used to link the other genes. The gene G_0 from table 1 is the linking gene and the whole chromosome encodes the expression (p * q) + (r / s) - (t + u).

Table 1. Expression encoded by multigenic chromosome

Gene	G_0	G_1	G_2	G_3
Symbols	+ -	* p q	/ r s	+ t u

These extra genes may also take part in the evolutionary process and evolve on the same rules as the other genes. Gene pool is formed from all combinations of operators and operands. From that pool some arbitrary genes are selected to construct a chromosome. Using different combinations of genes different chromosomes can be constructed. With 3 different genes and 2 different binary operator (3! + 2! -1) = 7 possible combinations (chromosome) can be generated. In general if there are m number of genes and n number of concatenating operators then (m! + n! -1) possible combination (Chromosome) can be generated. A more general scenario is obtained when the encoded expressions are made up of operands and operators of different types. The operators of each gene type are homogenous, meaning they all take arguments of the same type as inputs and they all return the argument of same type. The operators of each gene type are of the same type as the arguments the operators take. They may be simple symbols of required types or references to other genes where the aggregation operator also returns a compatible type. In the operand set there is also found the set of gene references that point to genes where operators return the compatible type. The multigenic chromosome in table 2 shows an expression containing:

- Context C_1- In linking gene G_0, operators that take Boolean arguments and return Boolean values
- Context C_2- in genes G_1, G_2 and G_3 – operators that take real arguments and return Boolean values
- Context C_3- in genes G_4, G_5, G_6 – operators that take real arguments and return real values

Table 2. Multigenic chromosome $(a<G_3)||(G_4>G_5)\&\&(b>c)$ with linking gene G_0

G_0	G_1	G_2	G_3	G_4	G_5	G_6
|| && G_1 G_2 G_3	< a G_6	> G_4 G_5	> b c	+ d f	- g h	* i j

The encoded expression is: $(a<G_3)||(G_4>G_5)\&\&(b>c)$ which further is expanded by dereferencing into (a<(i*j)) || ((d+f)>(g-h)) && (b>c) which is valid regarding the compatibility of types. Judging by the above three contexts are identified.

- The context C_1 of operators taking Boolean arguments and returning Boolean values; Operators={ ||, && } Operands ={ G_1, G_2, G_3}
- The context C_2 of operators taking real arguments and returning Boolean values; Operators = { <,>}, Operands = {a, b, c, d, f, g, h, i, j, G_4, G_5, G_6}
- The context C_3 of operators taking real arguments and returning real values; Operators = { +, -} Operands={ a, b, c, d, e, f, g}

This approach permits a linear representation of complex expressions.

4.2 Power Consumption

If average power consumption of each rekeying is 100 units and if rekeying number is expected as 100 times then 10000 units will be total power for rekeying in a sensor node. Each operator takes certain unit for execution.

- Operator '+' and '-' will take one unit because its execution takes one time unit.
- Operator '*' can be think of collection of several '+' operators. So power consumption for single '*' is equal to number of '+' operation to be performed.
- Operator '/' can be thinkof collection of several '-' operators. So power consumption for single '-' is equal to number of '-'operation to be performed.

But '*' and '/' and at least takes one time unit as an execution time so, consume at least one unit. Now '*' and '/' belongs to same priority level and has the highest priority among all operators. Operator '+' and '-' has the 2nd highest priority and both belongs to the same priority class. Logic operators have the lowest priority.

Each sensor node has equal amount of limited power primarily. This rekeying operation must be less power consuming in order to perform other jobs in sensor node with rest of the power. But a less power consuming rekeying operation produce a simple KGF which will easily be cracked. Thus, an appropriate KGF should be constructed by using following power consumption constraints given in equation 1.

$$PowerCon_L \leq PowerCon(Chromosome_i) \leq PowerCon_U \tag{1}$$

$$PowerCon_L = p_1 * g \tag{2}$$

$$PowerCon_U = p_2 * g + tolerence \tag{3}$$

where,

P_1 = average allowed minimum power consumption value of an operator for constructing not too simple chromosome.

P_2 = average allowed maximum power consumption value of an operator for constructing not too power consuming complex chromosome.

g = no. of genes.

Let $PowerCon(Chromosome_i)$ = (1 * 4) + (6 / 2) - (5 + 7)

Let P_1 and P_2 set as 4, *number of* g is 3 and *tolerance* is 6 then

$$(4*3) \leq PowerCon(Chromosome_i) \leq (4*3+6)$$

Power consumption of each gene depends on its operator. A single chromosome can be made of several genes. Different combination of gene can made several versions of chromosomes with different power consumption (takes different execution time). So those chromosomes violated power consumption constrained are ruled out.

4.3 Initial Population

In this phase after with an optimal power consumption constraints, chromosomes are arbitrarily generated using chromosome construction rules for creating chromosome pool.

4.4 Fitness Calculation

Fitness function is essential for producing quality solution. Using fitness function best parents set of genes (chromosome) are selected from pool of genes to produce good set of genes (chromosome). Proposed ECEEKO technique entropy function [2, 22] is used for fitness calculation of a chromosome. m number of genes and n number of linking operators may be produced at most (m!+n!-1) chromosomes but evaluation of distribution of this much of keys are impossible. Only f (f << (m!+n!-1)) keys are selected for evaluation. The set of f randomly selected keys are {$k_1, k_2, \ldots, k_f$}with probabilities ($p_1, p_2, \ldots, p_f$,).

$$\sum_{i=1}^{f} p_i, p_i \geq 0, i = 1,2,\ldots,f \tag{4}$$

$$Entropy(Gene\_Pool_i) = \sum_{i=1}^{f} p_i \log_2 \frac{1}{p_i} \tag{5}$$

Using equation (5) entrophy of a gene pool is calculated which is the fitness measure for that particular gene pool.For more uniform and unbiased key distribution maximum fitness value is required.

For example, consider genes G_1 =* 1 4, G_2 = / 6 2, G_3 = + 5 7 and G_4 = - 9 2 with 3 linking binary operators +, -, *. Using these 4 genes and 3 linking operator's total (4! + 3! -1) =29 possible valid chromosomes can be constructed in the following way.

- 4 genes G_1, G_2, G_3, G_4 can be linked 3! = 6 ways applying different combinations of 3 linking operators maintaining the gene order (G_1, G_2, G_3, G_4). So if there are m different linking operators then m! different chromosomes may be produced in this way. Table 3 shows 6 different such chromosomes.

Table 3. Shows 6 different chromosomes having same gene order.

$(G_1+G_2-G_3*G_4)$	$(G_1+G_2*G_3-G_4)$	$(G_1-G_2+G_3*G_4)$	$(G_1-G_2*G_3+G_4)$	$(G_1*G_2+G_3-G_4)$	$(G_1*G_2-G_3+G_4)$

- Using same linking operators +, -, * these 4 genes $G_{1,}$ $G_{2,}$ $G_{3,}$ G_4 can be linked with each other in 4!=24 different ways applying same order of linking operators but different gene order shown in table 4. If there are n different genes then n! different chrosomes can be constructed.

Table 4. Twentyfour different chromosomes having same order of linking operators but different order of genes.

$(G_1+G_2-G_3*G_4)$	$(G_2+G_1-G_3*G_4)$	$(G_3+G_2-G_1*G_4)$,	$(G_4+G_1-G_2*G_3)$,
$(G_1+G_2-G_4*G_3)$	$(G_2+G_1-G_4*G_3)$	$(G_3+G_2-G_4*G_1)$	$(G_4+G_1-G_3*G_2)$,
$(G_1+G_3-G_2*G_4)$	$(G_2+G_3-G_1*G_4)$	$(G_3+G_1-G_2*G_4)$,	$(G_4+G_2-G_1*G_3)$,
$(G_1+G_3-G_4*G_2)$	$(G_2+G_3-G_4*G_1)$	$(G_3+G_1-G_4*G_2)$,	$(G_4+G_2-G_3*G_1)$,
$(G_1+G_4-G_2*G_3)$	$(G_2+G_4-G_1*G_3)$	$(G_3+G_4-G_1*G_2)$,	$(G_4+G_3-G_1*G_2)$,
$(G_1+G_4-G_3*G_2)$	$(G_2+G_4-G_3*G_1)$	$(G_3+G_4-G_2*G_1)$,	$(G_4+G_3-G_3*G_1)$.

From the avove two processes, it is observed that if there are m number of genes and n numbers of linking operators then total possible chromosomes (different gene combination) will be (m! + n!)-1. Deduction of 1 because presence of 1 common gene in both the process i.e. $(G_1+G_2-G_3*G_4)$ in the above example. For each combination of genes and operators in particular gene pool entropy value is calculated. Example of entropy calculation is shown in the following.

- Chromosome 1: If linking gene G_0 = + - * G_1 G_2 G_3 G_4 then chromosome key value will be
  = (1*4) + (6 / 2) - (5 + 7) * (9-2) = 4+3-12*7 = -77 (as shown in table 5.)

Table 5. Chromosome with linking gene G_0 = + - * G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 2: If linking gene G_0 = + * - G_1 G_2 G_3 G_4 then chromosome key value will be
  = (1 * 4) + (6 / 2) * (5 + 7) - (9-2) = 4+3*12-7 = 33 (as shown in table 6.)

Table 6. Chromosome with linking gene G_0 = + * - G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
+ * - G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 3: If linking gene G_0 = * + - G_1 G_2 G_3 G_4 then chromosome key value will be
  = (1 * 4) * (6 / 2) + (5 + 7) - (9-2) = 4*3+12-7 = 17(as shown in table 7.)

Table 7. Chromosome with linking gene G_0 = * + - G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
* + - G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 4: If linking gene G_0 = * - + G_1 G_2 G_3 G_4 then chromosome key value will be
  = (1 * 4) * (6 / 2) - (5 + 7) + (9-2) = 4*3-12+7 = 7 (as shown in table 8.)

Table 8. Chromosome with linking gene G_0 = * - + G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
* - + G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 5: If linking gene G_0 = - + * G_1 G_2 G_3 G_4 then chromosome key value will be
  = (1 * 4) - (6 / 2) + (5 + 7) * (9-2) = 4-3+12*7 = 85 (as shown in table 9.)

Table 9. Chromosome with linking gene G_0 = - + * G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
- + * G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 6: If linking gene for G_0 = - * + G_1 G_2 G_3 G_4 then chromosome key value will be = (1 * 4) - (6 / 2) * (5 + 7) + (9-2) = 4-3*12+7 = -25 (as shown in table 10.)

Table 10. Chromosome with linking gene G_0 = - * + G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
- * + G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 7: If linking gene for G_0 = + - * G_1 G_2 G_4 G_3 then chromosome key value will be = (1 * 4) + (6 / 2) - (9 - 2) * (5 + 7) = 4+3-7*12 = 12 (as shown in table 11.)

Table 11. Chromosome with linking gene G_0 = + - * G_1 G_2 G_4 G_3

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_2 G_4 G_3	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 8: If linking gene for G_0 = + - * G_1 G_3 G_2 G_4 then chromosome key value will be = (1 * 4) + (5 + 7) - (6 / 2) * (9 - 2) = 4+12-3*7 = -5 (as shown in table 12.)

Table 12. Chromosome with linking gene G_0 = + - * G_1 G_3 G_2 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_3 G_2 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 9: If linking gene for G_0 = + - * G_1 G_3 G_4 G_2 then chromosome key value will be = (1 * 4) + (5 + 7) - (9 - 2) * (6 / 2)= 4+12-7*3 = -5 (as shown in table 13.)

Table 13. Chromosome with linking gene G_0 = + - * G_1 G_3 G_4 G_2

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_3 G_4 G_2	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 10: If linking gene for G_0 = + - * G_1 G_4 G_2 G_3 then chromosome key value will be = (1 * 4) + (9 - 2) – (6 / 2) * (5 + 7) = 4+7-3*12 = -25 (as shown in table 14.)

Table 14. Chromosome with linking gene G_0 = + - * G_1 G_4 G_2 G_3

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_4 G_2 G_3	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 11: If linking gene for G_0 = + - * G_1 G_4 G_3 G_2 then chromosome key value will be = (1 * 4) + (9-2) - (5 + 7) * (6 / 2) = 4+7-12*3= -25 (as shown in table 15.)

Table 15. Chromosome with linking gene G_0 = + - * G_1 G_4 G_3 G_2

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_4 G_3 G_2	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 12: If linking gene for G_0 = + - * G_2 G_1 G_3 G_4 then chromosome key value will be = (6 / 2) + (1 * 4) - (5 + 7) * (9 - 2) = 3+4-12*7 = -77 (as shown in table 16.)

Table 16. Chromosome with linking gene G_0 = + - * G_2 G_1 G_3 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_1 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 13: If linking gene for G_0 = + - * G_2 G_1 G_4 G_3 then chromosome key value will be = (6 / 2) + (1 * 4) - (9 - 2) * (5 + 7) = 3+4-7*12 = -77 (as shown in table 17.)

Table 17. Chromosome with linking gene G_0 = + - * G_2 G_1 G_4 G_3

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_1 G_4 G_3	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 14: If linking gene for G_0 = + - * G_2 G_3 G_1 G_4 then chromosome key value will be = (6 / 2) + (5 + 7) - (1 * 4) * (9 - 2) = 3+12-4*7 = -13 (as shown in table 18.)

Table 18. Chromosome with linking gene G_0 = + - * G_2 G_3 G_1 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_3 G_1 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 15: If linking gene for G_0 = + - * G_2 G_3 G_4 G_1 then chromosome key value will be = (6 / 2) + (5 + 7) - (9 - 2) * (1 * 4) = 3+12-7*4 = -13 (as shown in table 19.)

Table 19. Chromosome with linking gene G_0 = + - * G_2 G_3 G_4 G_1

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_3 G_4 G_1	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 16: If linking gene for G_0 = + - * G_2 G_4 G_1 G_3 then chromosome key value will be = (6 / 2) + (9 - 2) – (1 * 4) * (5 + 7) = 3+7-4*12 = -38 (as shown in table 20.)

Table 20. Chromosome with linking gene G_0 = + - * G_2 G_4 G_1 G_3

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_4 G_1 G_3	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 17: If linking gene for G_0 = + - * G_2 G_4 G_3 G_1 then chromosome key value will be = (6 / 2) + (9 - 2) – (5 + 7) * (1 * 4) = 3+7-12*4 = -38 (as shown in table 21.)

Table 21. Chromosome with linking gene G_0 = + - * G_2 G_4 G_3 G_1

G_0	G_1	G_2	G_3	G_4
+ - * G_2 G_4 G_3 G_1	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 18: If linking gene for G_0 = + - * G_3 G_2 G_1 G_4 then chromosome key value will be = (5 + 7) + (6 / 2) - (1 * 4) * (9 - 2) = 12+3-4*7 = -13 (as shown in table 22.)

Table 22. Chromosome with linking gene G_0 = + - * G_3 G_2 G_1 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_3 G_2 G_1 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 19: If linking gene for G_0 = + - * G_3 G_2 G_4 G_1 then chromosome key value will be = (5 + 7) + (6 / 2) - (9 - 2) * (1 * 4) = 12+3-7*4 = -13 (as shown in table 23.)

Table 23. Chromosome with linking gene G_0 = + - * G_3 G_2 G_4 G_1

G_0	G_1	G_2	G_3	G_4
+ - * G_3 G_2 G_4 G_1	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 20: If linking gene for G_0 = + - * G_3 G_1 G_2 G_4 then chromosome key value will be = (5 + 7) + (1 * 4) – (6 / 2) * (9 - 2) = 12+4-3*7= -5 (as shown in table 24.)

Table 24. Chromosome with linking gene G_0 = + - * G_1 G_2 G_3 G_4

G_0	G_1	G_2	G_3	G_4
+ - * G_1 G_2 G_3 G_4	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 21: If linking gene for G_0 = + - * G_3 G_1 G_4 G_2 then chromosome key value will be = (5 + 7) + (1 * 4) – (9 - 2) * (6 / 2) = 12+4-7*3= -5 (as shown in table 25.)

Table 25. Chromosome with linking gene G_0 = + - * G_3 G_1 G_4 G_2

G_0	G_1	G_2	G_3	G_4
+ - * G_3 G_1 G_4 G_2	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 22: If linking gene for G_0 = + - * G_3 G_4 G_1 G_2 then chromosome key value will be = (5 + 7) + (9 - 2) – (1 * 4) * (6 / 2) = 12+7-4*3=7 (as shown in table 26.)

Table 26. Chromosome with linking gene G_0 = + - * G_3 G_4 G_1 G_2

G_0	G_1	G_2	G_3	G_4
+ - * G_3 G_4 G_1 G_2	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 23: If linking gene for G_0 = + - * G_3 G_4 G_2 G_1 then chromosome key value will be = (5 + 7) + (9 - 2) – (6 / 2) * (1 * 4) = 12+7-3*4=7 (as shown in table 27.)

Table 27. Chromosome with linking gene $G_0 = + - * G_3 G_4 G_2 G_1$

G_0	G_1	G_2	G_3	G_4
$+ - * G_3 G_4 G_2 G_1$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 24: If linking gene for $G_0 = + - * G_4 G_1 G_2 G_3$ then chromosome key value will be = (9 - 2) + (1 * 4) – (6 / 2) * (5 + 7) = 7+4-3*12= -25 (as shown in table 28.)

Table 28. Chromosome with linking gene $G_0 = + - * G_4 G_1 G_2 G_3$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_1 G_2 G_3$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 25: If linking gene for $G_0 = + - * G_4 G_1 G_3 G_2$ then chromosome key value will be = (9 - 2) + (1 * 4) – (5 + 7) * (6 / 2)= 7+4-12 * 3= -25 (as shown in table 29.)

Table 29. Chromosome with linking gene $G_0 = + - * G_4 G_1 G_3 G_2$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_1 G_3 G_2$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 26: If linking gene for $G_0 = + - * G_4 G_2 G_1 G_3$ then chromosome key value will be = (9 - 2) + (6 / 2) – (1 * 4) * (5 + 7) = 7+3-4*12 =-38 (as shown in table 30.)

Table 30. Chromosome with linking gene $G_0 = + - * G_4 G_2 G_1 G_3$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_2 G_1 G_3$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 27: If linking gene for $G_0 = + - * G_4 G_2 G_3 G_1$ then chromosome key value will be = (9 - 2) + (6 / 2) – (5 + 7) * (1 * 4) = 7+3-12*4 =-38 (as shown in table 31.)

Table 31. Chromosome with linking gene $G_0 = + - * G_4 G_2 G_3 G_1$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_2 G_3 G_1$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 28: If linking gene for $G_0 = + - * G_4 G_3 G_1 G_2$ then chromosome key value will be = (9 - 2) + (5 + 7) – (1 * 4) * (6 / 2) = 7+12-4*3 = 7 (as shown in table 32.)

Table 32. Chromosome with linking gene $G_0 = + - * G_4 G_3 G_1 G_2$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_3 G_1 G_2$	* 1 4	/ 6 2	+ 5 7	- 9 2

- Chromosome 29: If linking gene for $G_0 = + - * G_4 G_3 G_2 G_1$ then chromosome key value will be = (9 - 2) + (5 + 7) – (6 / 2) * (1 * 4) = 7+12-3*4 = 7 (as shown in table 33.)

Table 33. Chromosome with linking gene $G_0 = + - * G_4 G_3 G_2 G_1$

G_0	G_1	G_2	G_3	G_4
$+ - * G_4 G_3 G_2 G_1$	* 1 4	/ 6 2	+ 5 7	- 9 2

After evaluation of these 29 different chromosomes 29 different values -77, 33, 17, 7, 85, -25, 12, -5, -5, -25, -25, -77, -77, -13, -13, -38, -38, -13, -13, -5, -5, 7, 7, -25, -25, -38, -38, 7, 7, get produced. Though these values are generated from same set of genes and operators but their different combinations, they are togatherly formed key set i.e. KEY = {-77, 33, 17, 7, 85, -25, 12, -5, -25, -13, -38, }. Now using equation (4) probabilities of selecting each key is calculated i.e. *p* = {1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11} and from the equation (5) entropy of this gene pool is calculated

$$Entropy(Gene\_Pool_i) = \sum_{i=1}^{f} p_i \log_2 \frac{1}{p_i}$$

In this way entropy of each gene pool is calculated. After that gene pool having maximum entropy value is selected as a KGF for a particular session. Now genes belonging to this best KGF get transmitted to the header nodes. Different header nodes can use different combination of these gene values for producing different key values.

4.5 Crossover Operation

The simple one point crossover proposed in [19, 20]. A chromosome is consisting of several genes and each gene belongs to a particular context (discussed in section 4.1). Table 34 shows a single chromosome with multiple contexts.

Table 34. Multigenic chromosome with multiple gene contexts

Context	C_1	C_2	C_2	C_2	C_3	C_3	C_3
Gene	G_0	G_1	G_2	G_3	G_4	G_5	G_6

Now, table 35 shows two multigenic chromosomes have shown made up of genes corresponding to the contexts where c_i denotes a gene of context i, crossover at gene 2 leads to the new chromosomes shown in table 36, which are chromosomes of the same gene structure.

Table 35. Multigenic chromosomes before crossover

Crossover Point

$\mathbf{C_1'}$	$\mathbf{C_2'}$	$\mathbf{C_2'}$	$\mathbf{C_2'}$	$\mathbf{C_3'}$	$\mathbf{C_3'}$	$\mathbf{C_3'}$
$\mathbf{C_1''}$	$\mathbf{C_2''}$	$\mathbf{C_2''}$	$\mathbf{C_2''}$	$\mathbf{C_3''}$	$\mathbf{C_3''}$	$\mathbf{C_3''}$

Table 36. Multigenic chromosomes after crossover

Crossover Point

$\mathbf{C_1'}$	$\mathbf{C_2'}$	$\mathbf{C_2''}$	$\mathbf{C_2''}$	$\mathbf{C_3''}$	$\mathbf{C_3''}$	$\mathbf{C_3''}$
$\mathbf{C_1''}$	$\mathbf{C_2''}$	$\mathbf{C_2'}$	$\mathbf{C_2'}$	$\mathbf{C_3'}$	$\mathbf{C_3'}$	$\mathbf{C_3'}$

In random position crossover when a crossover points falls inside a gene then given the genes G_i and G_{i+1} of the same context C_3, as shown in table 37 crossover at position 14 leads to the situation presented in table 38, which further conserves the validity of the chromosome.

Table 37 . Genes from a chromosomes involved in crossover at arbitrary position before the operation

Crossover Point

Gene		***Gene$_i$***				***Gene$_{i+1}$***		
Position	10	11	12	13	14	15	16	17
Symbol	+	a	b	-	*	d	e	f
Gene		***Gene$_i$***				***Gene$_{i+1}$***		
Position	10	11	12	13	14	15	16	17
Symbol	/	m	n	*	+	p	q	r

Table 38. Genes from chromosomes involved in crossover at an arbitrary position after the operation

Crossover Point

Gene		***Gene$_i$***				***Gene$_{i+1}$***		
Position	10	11	12	13	14	15	16	17
Symbol	+	a	b	-	+	**p**	**q**	**r**
Gene		***Gene$_i$***				***Gene$_{i+1}$***		
Position	10	11	12	13	14	15	16	17
Symbol	/	m	n	*	*	d	e	f

4.6 Mutation Operation

Single point mutation operation takes place in a context of a gene. At the mutation point the exchanged symbols must come from the corresponding sets of operands and operators. Consider the gene from table 39 where mutation point at 14 change the symbol and the new symbol comes from same context of the operator or operand set. In this example the symbol at positon 14 i.e "*" change to "+" which is belongs to the same context.

Table 39. Genes from chromosome involved in mutation at an arbitrary position before the operation

Mutation Point

Gene		***Gene$_i$***				***Gene$_{i+1}$***		
Position	10	11	12	13	**14**	15	16	17

Symbol	+	a	b	-	*	d	e	f

Table 40. Genes from chromosome involved in mutation at an arbitrary position before the operation

Gene		***Gene$_i$***			*Mutation Point*	***Gene$_{i+1}$***		
Position	10	11	12	13	**14**	15	16	17
Symbol	+	a	b	-	+	d	e	f

4.7 Second Phase

ECEEKO techniques scheme generates KGF satisfying following requirements:

- Less power consumption.
- Less memory usage for storing the set of genes and linking operators.
- Provides computation complexity for security.

ECEEKO scheme is divided into 3 parts :

- Sink node as a root.
- Header nodes as a intermediate nodes.
- Sensor nodes as a leaf node in the hierarchical tree structure.

All these nodes initially share a common key K_{share} which is used to initialize the encryption and discard K_{share} as soon as first rekeying is complete. After physical deployment each sensor nodes starts a timer. This ECEEKO scheme estimate the time after which a sensor node compromised and chooses a time bound parameter t much smaller than estimated time for rekeying purpose. In this ECEEKO hierarchical key generation strategy 3 types of key used for executing rekeying scheme in sink node, header nodes and sensor nodes respectively.

- Sink keys used for sink node and headers.
- Forward keys used among headers.
- Leaf keys used in sensor nodes and their respective headers.

All the above keys are formed using rearranging the genes with the help of linking operators. For generating leaf keys, header nodes arbitrarily select set of genes and rearrange the genes to generate random chromomes which then sends to the sensor nodes. Leaf nodes belongs to same header node use same sequenceof genes i.e. chromosome as a leaf key.

5. RESULTS AND ANALYSIS

In this section different experimental results and performance analysis of the ECEEKO technique have been done. Following is the parameters used in experiment:

- The initial population size is 30 (i.e. number of genes).
- the crossover rate is set at 0.8.
- The mutation rate is set at 0.01.

The ECEEKO approach has been compared with three existing techniques called KGF_RAND_1, KGF_RAND_2 and DDHV-D. In $KGF_RAND_1$and KGF_RAND_2 scheme 5 and 4 KGFs are randomly chosen respectively for its rekeying purpose. The DDHV-D scheme referred to in [3]. Simulation nvironment consist of 1 sink node, 10^3 header nodes, and 10^5 sensor nodes for constructing hierarchical wireless sensor network. In ECEEKO scheme the communication size can be calculated from the equation (6).

$$E_{k_i}(r_m)\,||\,KGF_i \oplus MAC_{k_i}(r_m) \tag{6}$$

Communication size =30*5+64=214 bits. Total cost: 214+ (30+120)/2=289 power units. Assumes ECEEKO scheme needs 100 power units for rekeying. The DDHV-D scheme Transfer A'= 2^6*10^5/10^3= 2^6*10^2 bits for rekeying where 2^6 is the key length. The power cost of DDHV-D is 2^6*10^2 *100/289=2215 power units. Assume it needs only 1 power unit for rekeying. Total cost: 2215+1=2216 power unit.
The parameters for the simulation process listed in table 41.

Table 41. Parameters used in simulation

	Schemes			
	KGF_RAND1	**KGF_RAND2**	**ECEEKO**	**DHDV-D**
Range of integers	1~100			
Power consumption for each rekeying	No constraint	No constraint	30~120	2216
Num of operators	4	6	7	-
Num of operands	5	7	7	-
KGF$_i$ length	4	6	7	-
Total genes	20	30	30	-

Amont of power remaining for the proposed ECEEKO and some existing KGF_RAND$_{1,}$ KGF_RAND$_2$ and DDHV-D scheme after each rekeying process is shown in figure 1. Proposed ECEEKO consume power in very stable manner where as KGF_RAND$_{1,}$ KGF_RAND$_2$ and DDHV-D scheme consumes the power very quickly and become powerless after a quite a few rekeyings.

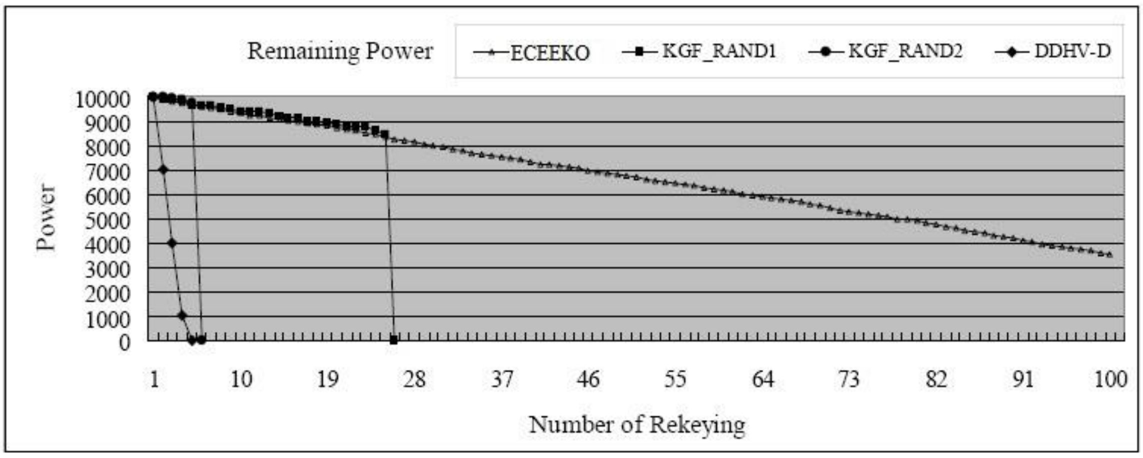

Figure1. Power vs. Number of rekeying chart for showing amount of power remaiming after each rekeying.

Power cost for each rekeying for the scmes ECEEKO, KGF_RAND$_{1,}$ KGF_RAND$_2$ and DDHV-D is shown in figure 2. From the chart it is observe that DDHV-D schme consume maximum power for each rekeying among these schmes and ECEEKO consumes minimum power.

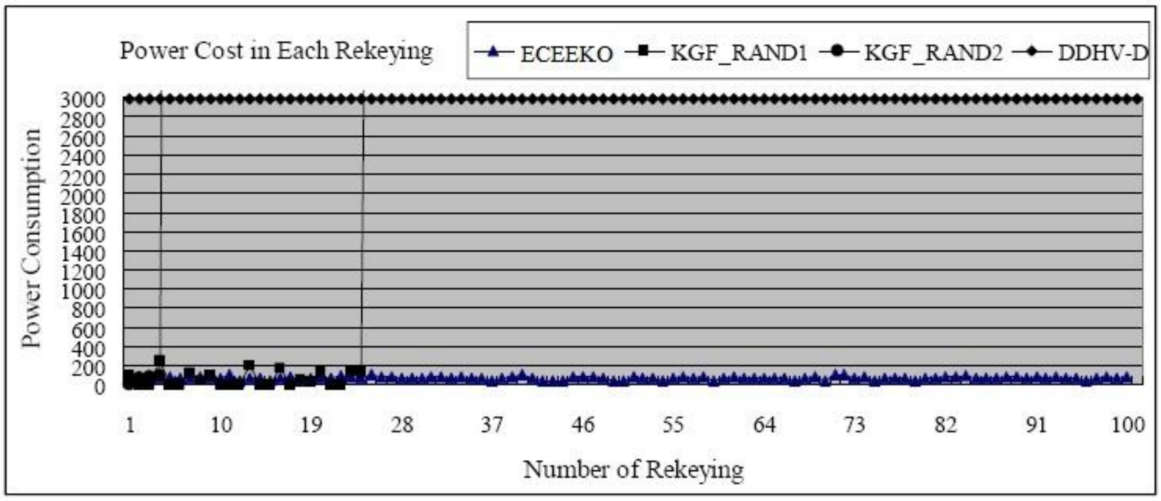

Figure 2. Power consumption vs number of rekeying chart for showing amount of power consume in each rekeying.

Entropy values for different set of genes are shown in figure 3. From the chart it is observed that more number of genes causes bigger entropy value. When number of genes about 52-56 the entropy value converge.

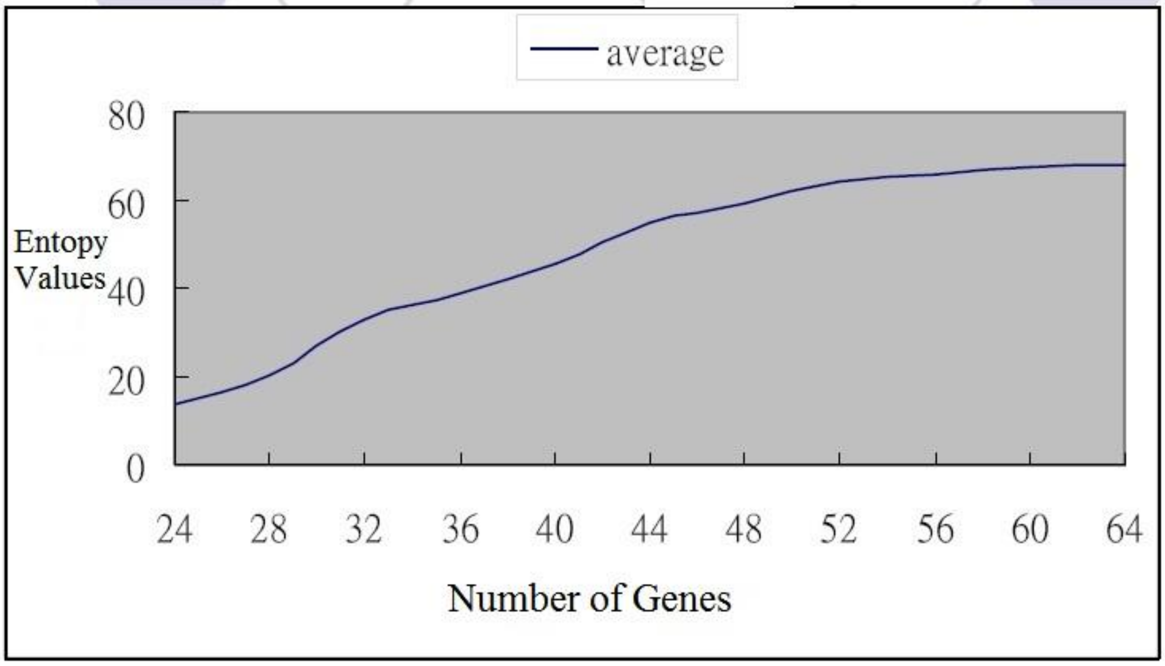

Figure 3. Number of genes vs entropy values

6. COMPLEXITY ANALYSIS

In this section space complexity of the ECEEKO technique has been analyzed in terms of memory usage. Proposed ECEEKO technique calculated the memory usage with the help of following equation (7).

$$(\log_2(p*q)+48)*p*q \tag{7}$$

Where,

$\log_2(p*q)$ is the length of series which is set as 8 bits. $(p*q)$ is the total number of genes i.e. 256 and 48 means that other package overhead. So, in ECEEKO technique (8+48)*256=14.336 kilobits to store 256 genes. In experiments of computation security, ECEEKO scheme is bounded on an attacker at most has to try (256! + 255! -1) possible way.Where 256 is the number of genes and 255 is the total number of linking operators for linking 256 genes. This is impractically for an attacker. In a word, ECEEKo scheme is a full connection scheme, but, the DDHV-D scheme is a partial connection scheme. Subsequence, ECEEKO memory usage is bigger than the DDHV-D scheme. However, the memory usage of ECEEKO fulfills the sensor nodes constraints.

7. CONCLUSION

In this paper, evolutionary computation guided energy efficient key has been proposed organization in wireless communication. First phase of this scheme deals with generation of good KGF where evolutionary computation is used by sink node for generating several KGF's for rekwying purpose in the second phase. Second phase deals with encoding of selected genes ans transmit them to header as well as sensor nodes for rekeying. From the experiment our opinion is that this ECEEKO scheme has some advantages that are:

- ECEEKO can be implemented in in low power sensor nodes.
- ECEEKO has a capability to find out KGF having low power consumption.
- ECEEKO also supports rekeying process dynamically.
- ECEEKO has a capability to control the power consumption in rekeying phase.
- ECEEKO scheme is very simple, efficient and secure.
- ECEEKO scheme has good space complexity compare to other existing schemes.
- ECEEKO provides uniform key distribution.

As a future scope of ECEEKO scheme, fitness function can be modified considering other parameters. New gene structures , better encoding of genes and different genetic operators can be consider for improvement of the ECEEKO scheme and thus further improve the performance of key organization scheme.

ACKNOWLEDGEMENTS
The author expressesed deep sense of gratitude to the Department of Science & Technology (DST) , Govt. of India, for financial assistance through INSPIRE Fellowship leading for a PhD work under which this work has been carried out, at the department of Computer Science & Engineering, University of Kalyani.

REFERENCES

[1] Chien-Lung Wang, Tzung-Pei Hong, Gwoboa Horng and Wen-Hung Wang, A GA-based Key-Management Scheme in Hierarchical Wireless Sensor Networks, *International Journal of Innovative Computing*, Volume 5,Number 12(A), pp. 4693-4702, ISSN1349-4198, December 2009.
[2] M. Ito and M. Tanaka, Localization of a Moving Sensor by Particle Filters, *International Journal of Innovative Computing*, Information and Control, vol. 4, no. 1, pp.165-174, 2008.
[3] W. Du, J. Deng, Y. S. Han and P. K. Varshney, A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge, *IEEE Transactions on Dependable and Secure Computing*, 2006.
[4] S. Camtepe and B. Yener, Key Distribution Mechanisms for Wireless Sensor Networks: A Survey, *Rensselaer Polytechnic Institute*, Troy, New York, Technical Report 05-07, 2005.
[5] S. Hahm, Y. Jung, S. Yi, Y. Song, I. Chong, and K. Lim, A Self-organized Authentication Architecture in Mobile Ad-hoc Networks, *International Conference on Information Networking*, pp.96-104, 2005.
[6] M. A. Moharrum, M. Eltoweissy, A Study of Static Versus Dynamic Keying Schemes in Sensor Networks, *ACM Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, 2005.
[7] A. Price, K. Kosaka, and S. Chatterjee, A Key Pre-distribution Scheme for Wireless Sensor Networks, *Wireless Telecommunications Symposium*, pp.253-260, 2005.
[8] M. Ramkumar, N. Memon, On the Security of Random Key Predistribution Schemes, *The Fifth Annual IEEE Information Assurance Workshop*, New York, 2004.
[9] H. Chan, A. Perrig, D. Song, Random Key Pre-distribution Schemes for Sensor Networks, *IEEE Symposium on Security and Privacy*, Berkeley, California, pp.197-213, 2003.
[10] M. Ramkumar, N. Memon, R. Simha, Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks, *IEEE Globe Telecommunication Conference*, San Fransisco, CA, 2003.
[11] S. Zhu, S. Setia and S. Jajodia, LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks, *ACM Conference on Computer and Communications Security*, 2003.
[12] L. Eschenauer and V.D. Gligor, A Key-Management Scheme for Distribution Sensor Networks, *ACM Computer and Comm. Security*, pp.41-47, 2002.

[13] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla and S. Lu, Adaptive Security for Multilevel Ad Hoc Network, *Wireless Communication and Mobile Computing*, 2002.
[14] G. Y. Lee; Y. Lee, Efficient Rekey Interval for Minimum Cost on Secure Multicast System Using Group Key, *IEEE Global Communications Conference*, Taipei, Taiwan, vol. 2, pp.1995-1999, 2002.
[15] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: Security Protocols for Sensor Networks, *ACM Annual International Conference on Mobile Computing and Networking*, pp.189-199, 2001.
[16] Y. Richard Yang, X. Steve Li, X. Brian Zhang, S. Simon Lam, Reliable Group Rekeying: A Performance Analysis, *ACM Annual Conference of the Special Interest Group on Data Communication*, vol. 31, Issue 4, 2001.
[17] A. Weimerskirch and G. Thonet, A Distributed Light-Weight Authentication Model for Ad-hoc Networks, *International Conference on Information Security and Cryptology*, 2001.
[18] E. Stajano, R. Anderson, The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks, *International Workshop on Security Protocols*, 1999.
[19] F. Herrera, M. Lozano and J. L. Verdegay, Fuzzy connectives based crossover operators to model genetic algorithms population diversity, *Fuzzy Sets and Systems*, vol. 92, no. 1, pp.21-30, 1997.
[20] David E. Goldberg, Genetic Algorithms in Search, *Optimization, and Machine Learning*, 1989.
[21] R. Blom, An Optimal Class of Symmetric Key Generation Systems, *Advances in Cryptology: Proceedings of EUROCRYPT*, pp.335-338, 1984.
[22] E. T. Jaynes, Information theory and statistical mechanics, *Physical Review*, vol. 106, pp.361-373, 1957.
[23] Rui Zhou , Hua Yang , A hybrid key management scheme for Heterogeneous wireless sensor networks based on ECC and trivariate symmetric polynomial, *Uncertainty Reasoning and Knowledge Engineering (URKE) 2011 International Conference on, sponsored by IEEE* pp. 251 – 255, Print ISBN: 978-1-4244-9985-4, 2011.
[24] Chen Chen; Zheng Huang; Qiaoyan Wen; Yajun Fan, A novel dynamic key management scheme for wireless sensor networks, *4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, pp. 549 – 552, Print ISBN: 978-1-61284-158-8, 2011.

BIOGRAPHY OF AUTHORS

Arindam Sarkar

INSPIRE Fellow (DST, Govt. of India) at the department of Computer Science & Engineering, University of Kalyani, MCA (VISVA BHARATI, Santiniketan, University First Class First Rank Holder), M.Tech (CSE, K.U, University First Class First Rank Holder). Total number of publications 12.

Jyotsna Kumar Mandal

M. Tech.(Computer Science, University of Calcutta), Ph.D.(Engg., Jadavpur University) in the field of Data Compression and Error Correction Techniques, Professor in Computer Science and Engineering, University of Kalyani, India. Life Member of Computer Society of India since 1992 and life member of cryptology Research Society of India. Dean Faculty of Engineering, Technology & Management, working in the field of Network Security, Steganography, Remote Sensing & GIS Application, Image Processing. 25 years of teaching and research experiences. Eight Scholars awarded Ph.D. one submitted and 8 are pursuing. Total number of publications 230.