

An Improved Cryptanalytic Attack on Knapsack Cipher using Genetic Algorithm

Poonam Garg and Aditya Shastri

Abstract—With the exponential growth of networked system and application such as eCommerce, the demand for effective internet security is increasing. Cryptology is the science and study of systems for secret communication. It consists of two complementary fields of study: cryptography and cryptanalysis. The application of genetic algorithms in the cryptanalysis of knapsack ciphers is suggested by Spillman [21]. In order to improve the efficiency of genetic algorithm attack on knapsack cipher, the previously published attack was enhanced and re-implemented with variation of initial entry parameters and results are compared with Spillman results. The experimental result shows that the efficiency of genetic algorithm attack on knapsack cipher can be improved with variation of initial entry parameters. The effect of initial entry parameters on genetic algorithm are also explored through experiments. Through extensive experiments and analysis it can be concluded that the by setting these initial entry parameter optimally we find a considerable improvement in the performance of genetic algorithm.

Keywords—Genetic Algorithm, Knapsack cipher, Key search

I. INTRODUCTION

THE demand for effective internet security is increasing exponentially day by day. Businesses have an obligation to protect sensitive data from loss or theft. Such sensitive data can be potentially damaging if it is altered, destroyed, or if it falls into the wrong hands. So they need to develop a scheme that guarantees to protect the information from the attacker.

Cryptology is at the heart of providing such guarantee. Cryptology is the science of building and analyzing different encryption and decryption methods. Cryptology consists of two subfields; Cryptography & Cryptanalysis. Cryptography is the science of building new powerful and efficient encryption and decryption methods. It deals with the techniques for conveying information securely. The basic aim of cryptography is to allow the intended recipients of a message to receive the message properly while preventing eavesdroppers from understanding the message.

Poonam Garg is presently Associate Professor of Computer Science at Institute of Management Technology, India.

Aditya Shastri is presently Professor of Computer Science & Director at Banasthali University, India

Cryptanalysis is the science and study of method of breaking cryptographic techniques i.e. ciphers. In other words it can be described as the process of searching for flaws or oversights in the design of ciphers.

One of first knapsack cipher was proposed by Merkle and Hellman [13] in 1978 which utilized a NP-complete problem for its security. The knapsack cryptosystem belongs to major categories of public/private key cryptosystem. The public/private key aspect of this approach lies in the fact that there are actually two different knapsack problems – referred to as the easy Knapsack and hard knapsack. The Markle-Hellman algorithm is based on this property. The hard knapsack becomes the public key. easy knapsack is the private key. The public key can be used to encrypt the message but can not be used to decrypt messages. The private key is used to decrypt the message.

The knapsack problem is formulated as follows. Let us assume the values M_1, M_2, \dots, M_n and the sum S are given. Let it be necessary to compute values b_1, b_2, \dots, b_n values, so that $S = M_1b_1 + M_2b_2 + \dots + M_nb_n$. The values of coefficient b_i can be equal 0 or 1. The 1 value shows that object will fit into the knapsack, 0 values will not in the knapsack.

The Markle-Hellman knapsack cipher encrypts a message as a knapsack problem. The plaintext block transforms into binary string (the length of block is equal number of elements in knapsack sequence). One value determines that an element will be in target sum. This sum is a ciphered message. Table I shows an example of solving the knapsack problem for the entry numbers sequence: 1 3 6 13 27 and 52.

TABLE I
EXAMPLE OF KNAPSACK ENCRYPTION

Plaintext	Knapsack sequence	Ciphertext
1 1 1 0 0 1	1 3 6 13 27 52	$1+3+6+52= 62$
0 1 0 1 1 0	1 3 6 13 27 52	$3+13+27 = 43$
0 0 0 0 0 1	1 3 6 13 27 52	$52 = 52$

Easy knapsacks have a sequence of numbers that are superincreasing - that is, each number is greater than the sum

of previous numbers: $a_i > \sum_{j=1}^{i-1} a_j$ for $i=2, \dots, n$ (where a_i

is i -th element of the sequence). For example $\{1,3,6,13,27,52\}$ is a superincreasing sequence but $\{1,3,4,9,15,25\}$ is not. The knapsack solution with the superincreasing sequence proceeds as follows. The target sum

is compared with a greatest number in the sequence. If the target sum is smaller, than this number, the knapsack will not fill, otherwise it will. Then the smaller element is subtracted from the target sum, and the result of the subtraction, is compared with next element. Such operation is done until the smallest number of sequence is reached. If the target sum is reduced to 0 value, than solution exists. In other case solution doesn't exist. For example, consider a total knapsack target sum is 70 and the sequence of weights of {2, 3, 6, 13, 27, and 52}. The largest weight, 52, is less then 70, so 52 are in the knapsack, Subtracting 52 from 70 leaves 18. The next number 27 is greater than 18, so 27 is not in the knapsack. The next weight 13 is less than 18, so 13 is in the knapsack. Subtracting 13 from 18 leaves 5. The next weight, 6, is greater than 5, so 6 are not in the knapsack. Continuing this process will show that both 2 and 3 are in the knapsack and the total weight is brought to 0, which indicates that a solution has been found. The plaintext that resulted from a ciphertext value of 70 would be 110101. The superincreasing knapsack is easy to decode, which means that it does not protect the data. Anyone can recover the bit pattern from the target sum for a superincreasing knapsack if the elements of the superincreasing knapsack are known.

Markle and Hellman suggested that such a simple knapsack be converted into a trapdoor knapsack which is difficult to break. The algorithm work as follows :

1. Select a simple knapsack sequence. Elements make a superincreasing $A'=(a'_1 + a'_2 \dots a'_n)$
2. Select an integer value m greater than sum of all elements of superincreasing sequence.
3. Select another inter w that the $\text{gcd}(m,w)=1$, that is number m and w are reciprocally prime.
4. Find the inverse of the $w \text{ mod } m - w^{-1}$
5. Construct the hard knapsack sequence $A=wA' \text{ mod } m$ i.e. $a_i=w a'_i \text{ mod } m$

The trapdoor sequence A could be published as a public key (encryption key). The private (secret) key for this cipher consists of a simple knapsack sequence A' , so-called trapdoor, values m, w, w^{-1}

The encoding is done as follows. The message divides into n bits block (each block contains as many element as simple knapsack sequence). Values in the message block shows that the element will be in the target sum. The target sum of each block is ciphertext.

The decoding consists of the following. Each number of the ciphered message is multiplied through $w^{-1} \text{ mod } m$ and the result of this operation is plaintext. The application of genetic algorithms in the cryptanalysis of knapsack ciphers is suggested by Spillman [21].

II. GENETIC ALGORITHM

The genetic algorithm is based upon Darwinian evolution theory. The genetic algorithm is modeled on a relatively

simple interpretation of the evolutionary process, however, it has proven to a reliable and powerful optimization technique in a wide variety of applications. Holland [9] in 1975, was first proposed the use of genetic algorithms for problem solving. Goldberg and Dejong [6] were also pioneers in the area of applying genetic processes to optimization. Over the past twenty years numerous application and adaptation of genetic algorithms have appeared in the literature.

During each iteration of the algorithm, the processes of selection, reproduction and mutation each take place in order to produce the next generation of solution.

Genetic Algorithm begins with a randomly selected population of chromosomes represented by strings. The GA uses the current population of strings to create a new population such that the strings in the new generation are on average better than those in current population (the selection depends on their fitness value).

Three processes which have a parallel in biological genetics are used to make the transition from one population to the next (selection, crossover, and mutation) see Fig. 1

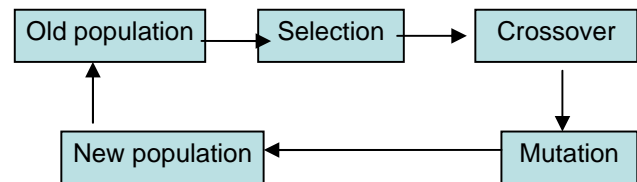


Fig. 1 The basic genetic algorithm cycle

The selection process determines which string in the current will be used to create the next generation. The crossover process determines the actual form of the string in the next generation. Here two of the selected parents are paired. A fixed small mutation probability is set at the start of the algorithm.

III. OBJECTIVE OF THE STUDY

Cryptanalytic attack on Knapsack cipher belongs to the class of NP-hard problem. The cryptanalytic attack on knapsack cipher using genetic algorithm exists in the literature [21]. Due to the constrained nature of the problem, this attack reaches to varying levels of performance optimization. Therefore, this paper is looking for a new solution that further improves the robustness against cryptanalytic attack with high effectiveness. So previously published attack [21] was enhanced and re-implemented with variation of initial entry parameters and results are compared with Spillman results.

The objective of the study is:

- To determine the efficiency of genetic algorithm on the cryptanalysis of knapsack cipher.
- Comparison of our result with Spillman result.
- To determine the effect of initial entry parameter on genetic algorithm

IV. METHODOLOGY

Fitness function

Spillman [21] proposed the fitness measure given in Equation 1.

$$\text{Fitness} = \begin{cases} 1 - \left(\frac{|T \text{ arg et} - \text{Sum}|}{\text{Target}} \right)^{\frac{1}{2}} & \text{if Sum} \leq \text{Target} \\ 1 - \left(\frac{|T \text{ arg et} - \text{Sum}|}{\text{MaxDiff}} \right)^{\frac{1}{6}} & \text{if Sum} > \text{Target} \end{cases} \quad (1)$$

Let $M = \{m_1, m_2, \dots, m_n\}$, $m_i \in \{0, 1\}$ be an arbitrary solution and the public key

$$A = \{a_1, a_2, \dots, a_n\}$$

$$\text{Sum} = \sum_{j=1}^n a_j m_j$$

$$\text{Target} = \sum_j a_j'$$

$$\text{FullSum} = \sum_{j=1}^n a_j$$

$$\text{MaxDiff} = \max \{ \text{Target}, \text{FullSum} - \text{Target} \}$$

Cryptanalytic attack on knapsack cipher using genetic algorithm

Spillman[21] suggested genetic algorithm to solve the knapsack problem. Fig. 2 shows the Markle-Hellman cryptosystem and cryptanalysis by means of genetic algorithm.

The cryptanalysis starts from cipher text, which has an integer form. Each number represents a target sum of hard knapsack problem. The goal of the genetic algorithm is to translate each number into the correct knapsack, which represents the ASCII code for the plaintext characters.

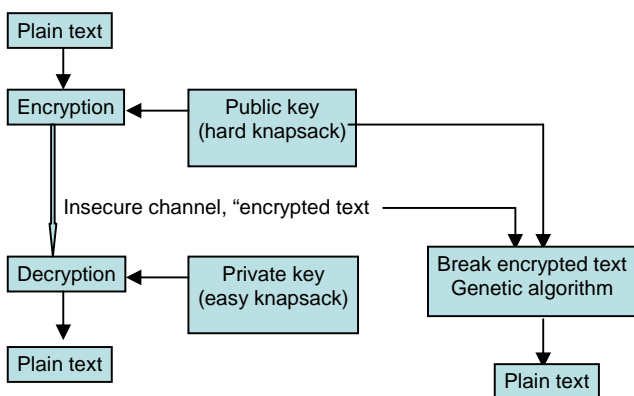


Fig. 2 Markle-Hellman cryptosystem and cryptanalysis by means of Genetic Algorithm

In this paper before further experiments the following initialization conditions are used for creating the initial population. Spillman’s work does not give any information about the size of population, crossover probability and mutation probability.

Encoding

The following restriction have been made for encoding

- (1) Only the ASCII code will be encrypted.
- (2) The superincreasing sequence will have 8 elements; these number of elements guarantee that each character has a unique encoding (There are 256 ASCII codes and 8 elements length will allow to encrypt 2^8 characters)
- (3) Plaintext has not more then 100 character length.

Initialization

A random population of chromosomes (binary string 0’s and 1’s) is generated. The size of the population has range in between 10 to 100. The number of bits in each chromosome is equal to the number of elements key (i.e. 8).

Evaluation

Based on the fitness function given in equation 1 the fitness value evaluates how the given sum is close to the target value for the knapsack. The value of the fitness function should be in the range of 0 to 1. Fitness value 1 indicates an exact match with the target sum for the knapsack. If the value of sum is greater then targets then it have a lower fitness value of chromosome, in this way it produces the infeasible solution. If the value of sum is less then target then it will produce a high fitness value and produce feasible solutions. Feasible solutions have a greater chance of being followed by the algorithm.

Selection

The important part of algorithm is selection of a new population. The convergence of the algorithm can be delayed or speed up it depends on how the criterion of selection defined. Two selection methods are applied here: elitist method and classical method. Hence, the 25% the best chromosome with parents population move to next population, the rest 75% choose after classical methods. The best chromosome of each population will preserve and not used in crossover operation or mutation process. Therefore, the results will not make it worse in the next population.

Crossover

The one-point crossover operation applied in the algorithm.

Mutation

The mutation process moves between two random points. The mutation probability has to be small. The mutation helps to prevent the algorithm from being stuck in a local optimal point.

Stop (Terminating) condition

After executing the above mentioned steps a new generation is created and the steps are repeated until the stop condition is reached. Two variants of stop conditions are applied. In the variant-I, the algorithm will stops when the fitness function reaches to the value 1. In the variant - II the algorithm will stop either the fitness function reaches to the value 1 or generates 250 populations.

Fig. 3 describes the complete steps of genetic algorithm that utilizes in the attack of knapsack cipher.

1. A random population of chromosomes (binary string 0's and 1's) is generated. The size of the population has range between 10 and 100. The number of bits in each chromosome is equal to the number of elements key (that is 8).
2. A fitness value for each chromosome in the population is determined with the help of equation 1.
3. The new population is created.
4. The crossover operation is made.
5. The mutation process is executed.
6. Repeat step 2 for the new population of chromosomes.

Fig. 3 Genetic algorithm attack on knapsack cipher

VI. RESULTS AND DISCUSSIONS

The experiments performed for this study consists of two main parts. The first part consists of re-implementation of cryptanalytic attack with variation of entry parameter and comparison of own results and Spillman's result. And second part consists of the effects of initial entry parameters (crossover, mutation and size of population) on genetic algorithm.

PART I

All experimental result for the genetic algorithms was generated with 10 runs per data point using 'C' language. Genetic algorithm attack is run with variant-I & variant-II for each target sum mentioned in Table II. Each attack is run 10 times with constant entry parameters i.e. size of population is 100, crossover probability is 0.90, and mutation probability is 0.25 and then results are averaged.

The 8-elements (Spillman used 15 elements) sequence of hard knapsack problem (21031 63093 16371 11711 23422 58555 16615 54322) is used to encode 8 bits ASCII code. This sequence has been created from superincreasing sequence (1 3 7 13 26 65 119 267), u equal to 65423 and w integer equal 21031 ($w^7 = 5363$). The MACRO word has been encrypted.

TABLE II
ENCRYPTION BY KNAPSACK

Character	ASCII Code	Target sum (ciphertext)
M	10110010	65728
A	10000010	37646
C	11000010	100739
R	01001010	103130
O	11110010	128821

The Tables III and Table IV show the experimental results with variant -I (P is population with search result, % is percentage of search space) and variant-II (P is population with search result; % is the percentage of search space, E is the Error i.e. difference between target sum and find sum) as a stop a condition.

Table III shows that, the solution with variant -I always gives correct solution because the population is generated till the result is not erroneous. On an average 115 populations are enough for reaching to the correct results. Table IV with variant-II shows that 43 populations are enough but the solutions can be incorrect because of burdened error. The Table II shows the result of encoding.

TABLE III
EXPERIMENTAL RESULTS WITH VARIANT - I

Char	Exp. 1		Exp. 2		Exp. 3		Exp. 4		Exp. 5		Avg	
	P	%	P	%	P	%	P	%	P	%	P	%
M	13	47.7	2	29.1	68	49.8	1	24.4	1	25.8	17	35.3
A	226	70.5	67	53.7	1	26.3	1	24.4	265	70.5	112	49.1
C	173	66.2	48	55.6	1	27.8	279	73.4	853	95.1	271	63.6
R	290	70.1	4	36.5	108	68.4	44	51.2	1	27.3	89.4	50.7
O	2	25.6	1	26.6	210	62.3	1	25	222	63.5	87.2	40.6
Sum											115	47.9

TABLE IV
EXPERIMENTAL RESULTS WITH VARIANT –II

Char	Exp.1			Exp.2			Exp.3			Exp.4			Exp.5			Avg.		
	P	%	E	P	%	E	P	%	E	P	%	E	P	%	E	P	%	%E
M	3	1.5	0	124	62	0	5	2.5	0	67	33.5	0	143	71.5	2635	68.4	34.2	20
A	65	32.5	0	1	0.5	2513	88	44	244	7	3.5	2513	7	3.5	244	33.6	16.8	80
C	1	0.5	0	67	33.5	0	162	81	0	2	1	0	115	57.5	0	69.4	34.7	0
R	1	0.5	244	132	66	0	1	0.5	0	1	0.5	2391	82	41	0	43.4	21.7	40
O	1	0.5	0	1	0.5	0	1	0.5	0	5	2.5	0	2	1	30229	2	1	20
Sum																43.32	21.7	32

The results shown in Table III are compared and analyzed with Spillman's results (Table V). Spillman's algorithm always gives correct results similar to results as obtained by us (Table III).

TABLE V
SPILLMAN'S RESULTS

Character	#chromosome	% of search space
M	810	2.0
A	80	0.2
C	1860	6.0
R	460	1.0
O	650	.1
Average	650	1.9

In our experiment average 115 populations gives the correct result and the average percentage of search space is near 50%. The Spillman's algorithm (Table V) searches on average less than 2% of the space. The divergence of the result is explained that the area of possible results in Spillman's work is 2^{15} i.e. 32678 and in our work it is 2^8 . The convergence of results is shown in Fig. 4.

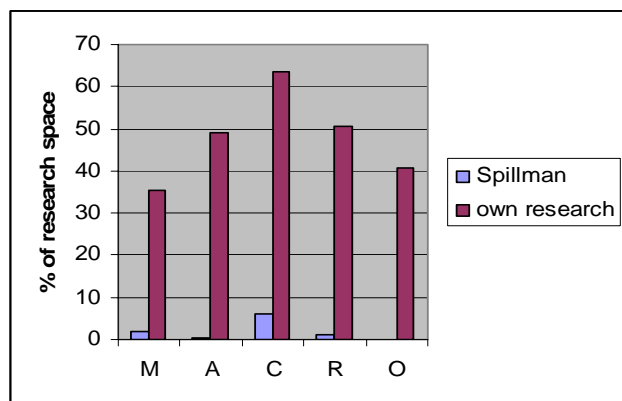


Fig. 4 The comparison of own result with Spillman's result

These results indicate that Spillman's results were distinctly suboptimal and the variation of initial entry parameters could readily improve the result.

PART II

Genetic algorithm work very efficiently, when good value for entry parameter such as population size, crossover rate, mutation rate is chosen. It is very beneficial, when these parameters are set optimally, because the genetic algorithm will yield better percentage of search space, in comparatively lesser number of generations. It is generally recognized the efficiency of genetic algorithm can be improved by variation of the initial entry parameter.

In this section effects of initial entry parameter like size of population, crossover probability & mutation probability on genetic algorithm are explored. By setting these parameters optimally we find a considerable improvement in the performance of the genetic algorithm.

Test I – Effect of Population size

Population is defined as an array of individual solutions. Thus the Size of population in important parameter that is to be considered for the success of the GA. If the population size is too small, then an insufficient number of samples are sampled and would not yield in the best possible solution. If the population size is too large, the algorithm becomes inefficient as more number of tests is performed than necessary for each generation.

In order to see the effect of population size on the number of generations and percentage of search space 6 tests are performed. All tests for the genetic algorithms were generated with 10 runs per data point using 'C' language. Genetic algorithm attack is run with variant-I & variant-II for each target sum mentioned in Table II. Each attack is run 10 times with varied population size from 10 to 100, constant crossover probability is 0.9 and mutation probability is 0.25 and then results are averaged.

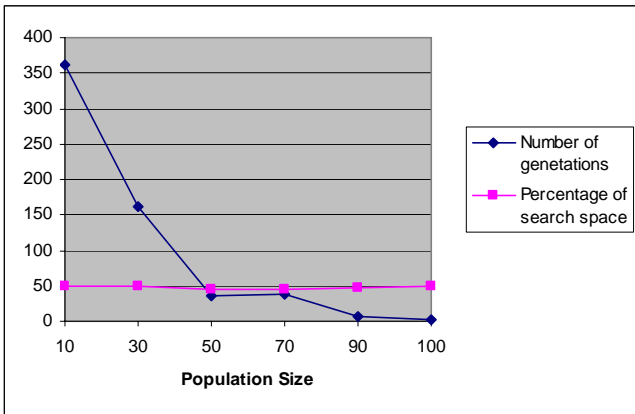


Fig. 5 Effect of size of population using variant-I

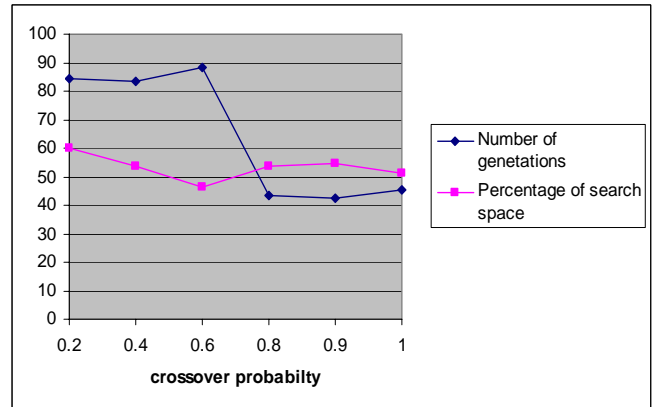


Fig. 7 Effect of crossover rate using variant-I

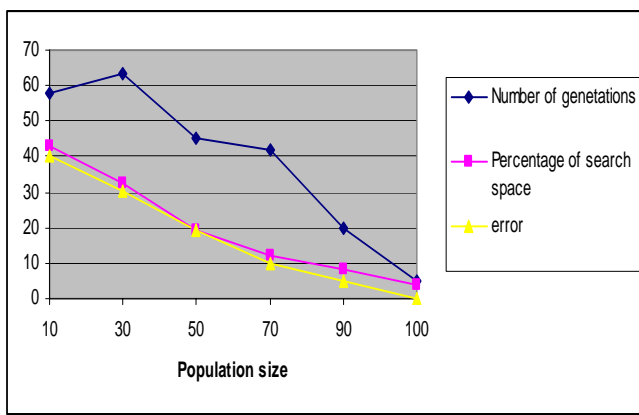


Fig. 6 Effect of size of population using variant-II

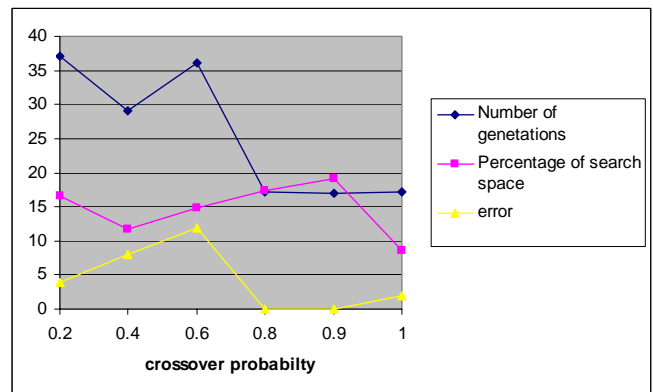


Fig. 8 Effect of crossover rate using variant-II

It can be observed from Fig. 5 & 6, the effect of size of population is inversely proportional to the number of generations. In other words if we are increasing the size of population then the number of generations are decreased. But it doesn't influence on the percentage of search space see Fig. 5. In variant-II, also when we are increasing the size of population then the number of generations and percentage of search space are decreasing. As Fig. 6 shows that number of erroneous solutions drops when the population is 100. So an optimal solution considered was a population of size 100.

Test II - Effect of Crossover rate

In order to see the effect of crossover rate on the number of generations and percentage of search space 6 tests are performed. All tests for the genetic algorithms were generated with 5 runs per data point using 'C' language. Genetic algorithm attack is run with variant-I & variant-II for each target sum mentioned in Table II. Each attack is run 10 times with varied crossover rate between .2 to 1, constant population size 100, constant mutation probability is 0.25 and then results are averaged. Fig. 7, shows the setting values of the number of generations and percentage of search space using variant -I. Fig. 8, shows the setting values of the number of generations, percentage of search space, number of errors in the solution using variant -II.

It can be seen from the Fig. 7 & 8, that a value of .8 - .9 seem to be good choice for cryptanalysis of knapsack cipher. When the crossover rate is set to low values (.2), the percentage of search space is higher in the initial stage but it eventually settles down the at low percentage search space. When the crossover rate is set to high value (.9 - 1), the genetic algorithm settles down in a lower percentage of search space, indicating that too high or too low the crossover rate will not yield optimal result. When the crossover rate of .8 to .9 is used, the genetic algorithm response and settles in the higher search space, in less number of generations, and number of erroneous solutions drops. The crossover rate was thus set to .9 for further analysis, as this gives higher percentage of search space in less number of generations and without erroneous solutions.

Test III - Effect of Mutation rate

In order to see the effect of mutation rate on the number of generations and percentage of search space 6 tests are performed. All tests for the genetic algorithms were generated with 10 runs per data point using 'C' language. Genetic algorithm attack is run with variant-I & variant-II for each target sum mentioned in Table II. Each attack is run 10 times with varied mutation rate between .01 to 1, constant population size 100, constant crossover probability is 0.9 and then results are averaged. Fig. 9, shows the setting values of

the number of generations and percentage of search space using variant -I. Fig. 10, shows the setting values of the number of generations, percentage of search space and number of errors in the solution using variant -II ,

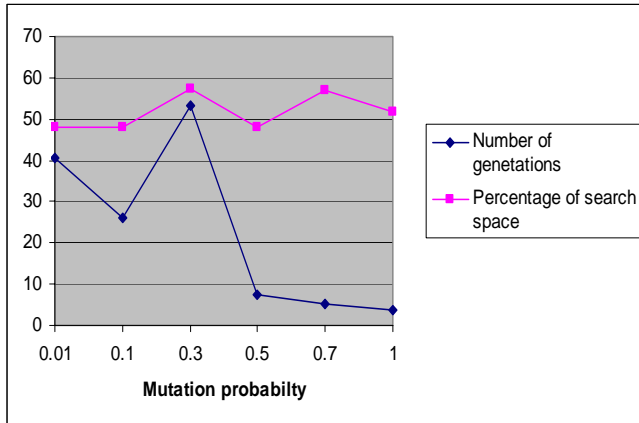


Fig. 9 Effect of mutation rate using variant-I

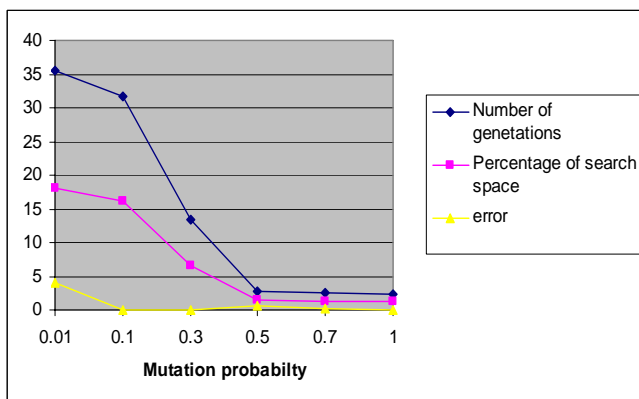


Fig. 10 Effect of mutation rate using variant-II

Despite of the fact that mutation is often regarded as a secondary operator to crossover. It can be seen here that it can produce high rates of leaning, if mutation rate is right. It can be observed from Fig. 9 & 10, when the mutation rate is very low such as .001, the percentage of search space is higher in the initial stage but it eventually settles down the at low percentage search space. When the mutation rate is set to high value (.3 - 1), the genetic algorithm settles down in a lower percentage of search space, indicating that too high or too low the mutation rate will not yield optimal result. When the mutation rate is in between of .1 - .3 is used, the genetic algorithm response and settles in the higher search space, in less number of generations, and number of erroneous solutions drops. The mutation rate was thus set to .25 for further analysis, as this gives higher percentage of search space in less number of generations and without erroneous solutions.

VII. CONCLUSION

This paper presents the genetic algorithm attack on the knapsack cipher. This paper also indicates that the efficiency of genetic algorithm attack on knapsack cipher can be improved by variation of initial assumptions like mutation, crossover operation and size of population. The great size of population, high crossover probability and a small mutation probability are the most optimal arrangements for GA. The results are worse when the size of population and crossover operator decreases, and when the coefficient mutation increases. Wrong coefficients destroy the chromosomes that are well-selected. In our experiment average 115 populations gives the correct result and the average percentage of search space is near 50%. The Spillman's algorithm searches on average less than 2% of the search space. Experimental results indicate that Spillman's results [21] were distinctly suboptimal and the variation of initial assumptions could readily improve the result. The parameters for the genetic algorithm were carefully set, by performing a number of trial runs. So it can be concluded that starting the genetic algorithm with great size of population, high crossover probability and small mutation rate can increase the efficiency and performance of genetic algorithm

The genetic algorithm offers a powerful tool for the cryptanalysis of knapsack cipher.

REFERENCES

- [1] G Alfred J. "Handbook of Applied Cryptography", CRC, 1997.
- [2] Clark A., ED Dawson and Helen Bergen, Combinatorial Optimization And The Knapsack Cipher, Cryptologia, vol. 20, no. 1, pp.85-93, Jan 1996.
- [3] Chor ,B. and Rivest R.L, A knapsack type public-key cryptosystem based on arithmetic in finite fields. IEEE Trans. Inform Theory 34(5)(1988)901-909.
- [4] B Davis, L. , "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York ,1991.
- [5] Frank R., Comments on "Cryptanalysis of Knapsack Cipher Using Genetic Algorithm". Cryptologia, vol. XVIII, no. 2, April 1994, pp. 153-154.
- [6] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, 1989.
- [7] Garg P., Genetic Algorithm & Tabu Search attack on monoalphabetic cipher, BIMC proceeding of Business Information Management Conference -2005, 322-328.
- [8] Garg P., Genetic Algorithm, Tabu Search & Simulated Annealing Attack on Transposition Cipher, proceeding of third AIMS International conference on management at IIMA - 2006, 983-989 1975.
- [10] Matoušek, R. , Hill Climbing and 0/1 Knapsack Problem, In Proceeding of Mendel 2002 Conference, Brno, Jul 2002
- [11] Matoušek, R., 2002, GA-HC: A Hybrid Genetic Algorithm, Fuzzy Colloquium, Zittau 2002
- [12] Methew, R.A.J. (1993, April), The use of genetic a algorithms in cryptanalysis, *Cryptologia*, 7(4), 187-201.
- [13] Merkle R. C, Martin E. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. IEEE transaction on Information Theory, vol. IT-24, 1978, pp. 525-530.
- [14] Michalewicz, Zbigniew. GENETIC ALGORITHMS + DATA STRUCTURES = EVOLUTION PROGRAM, 3rd Edition, Springer Verlag, 1996.
- [15] Odlyzko A. M., Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme. IEEE Transactions on Information Theory, IT-30, 1984, pp. 594-601. Available at <http://www.research.att.com/~amo/doc/arch/knapsack.attack.ps>

- [16] Odlyzko A. M.. The Rise and Fall of Knapsack Cryptosystems. In Carl Pomerance, editor, *Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics*, vol. 42. American Mathematics Society, Providence, RI, 1990, pp. 75-88.
Available at
<http://www.research.att.com/~amo/doc/arch/knapsack.survey.ps>
- [17] Schneier B. , "*Applied Cryptography*", 2nd edition , John Willy & Sons, Newyork, 1996.
- [18] Shamir A. A Polynomial-time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. Proceedings of the IEEE Symposium on Foundations of Computer Science. IEEE, New York, 1982, pp. 145-152.
- [19] Shamir A., A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. In David Chaum, Ronald L. Rivest, Alan T. Sherman. editors, *Advances in Cryptology – CRYPTO '82*. Plenum, New York, 1983.
- [20] Shamir. A., A Polynomial-time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. *IEEE Transactions on Information Theory*, vol. IT-30, no. 5, September 1984, pp. 699-704.
- [21] Spillman R., Cryptanalysis of knapsack ciphers using genetic algorithms, *Cryptologia*, 17(4):367–377, October 1993.
- [22] Yaseen, I.F.T., & Sahasrabuddhe, H.V. (1999). A genetic algorithm for the Cryptanalysis of Chor-Rivest knapsack public key cryptosystem (PKC). In *Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications*, pp. 81-85, 1999.

Poonam Garg is presently Associate Professor of Computer Science at Institute of Management Technology, India. She received her Master of Computer Applications in 1991 from Bansthali Vidyapith, India before completing her Ph.D. at Banasthali Vidyapith, India in 2006. Her research interests are in Network Security, Cryptography, Network planning. She has published various research papers in international/national journal and conferences.

Aditya Shastri is presently Professor of Computer Science & Director at Banasthali University. He received his M.Sc. (Tech.) Computer Science & M.Sc. (Hons.) Maths degree from BITS-Pilani, India before completing his Ph.D. at Massachusetts Institute of Technology, Cambridge USA in 1990. His research interests are in Discrete Mathematics, Graph Theory and Mobile Computing. He has written more than 50 research papers in journals of repute and authored 5 textbooks. Of late, he is engaged in the development of Banasthali University as its chief executive and chief academic officer.