Journal of Computer Science 7 (11): 1633-1638, 2011
ISSN 1549-3636

Reduction of Key Search Space of Vigenere Cipher Using Particle Swarm Optimization

[1]Ganapathi Sivagurunathan and [2]T. Purusothaman
[1]Faculty of Computer Science and Engineering, Anna University, Coimbatore, Tamilnadu,
[2]Faculty of Computer Science and Engineering Government College of Technology,
Coimbatore, Tamilnadu, India

Abstract: Problem statement: With the demand for effective network security is increasing, it becomes necessary to find the strength and weaknesses of the existing cryptographic methods. Vigenere cipher, a classical cipher is analyzed for its strength against a cipher only attack. **Approach:** The cipher texts so selected were of various sizes up to 1 Kb. A biologically inspired algorithm, Particle Swarm Optimization (PSO) was applied to the problem of crypt analyzing the Vigenere cipher. PSO was an optimization technique and its used on the problem of optimizing the fitness function designed for Vigenere cipher was performed. **Results:** It was seen that PSO is able to find the keyword employed and the other possible combinations for the keyword. **Conclusion:** PSO is better than genetic algorithm to solve Vigenere cipher and can be used to find the keyword with lesser size.

Key words: Network security, swarm optimisation, keyword employed, cipher text, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), vigenere cipher, english alphabets, plain text

INTRODUCTION

Cryptanalysis is the process of finding the Keyword employed for enciphering a plaintext and hence using the keyword found, finding the Plaintext of the given cipher text. Cryptanalysis is required to find the strengths and weaknesses of the cipher method employed and if the method is found to be vulnerable to cipher attacks. If the method employed is not strong against various attacks then it becomes necessary to reinforce the methodology employed (Al-Saidi and Said, 2009) or to find an alternate ciphering method. A symmetric cipher is the one which employs the same keyword for enciphering and deciphering. Such a simple classical symmetrical cipher is Vigenère cipher.

Several methods have been employed for the cryptanalysis of classical ciphers. The application of Genetic Algorithm (GA) to Vigenère cipher (Purusothaman *et al*., 2009) was performed and it was shown that GA is capable of breaking the Vigenère cipher. A fitness function was employed to find the probable keyword and then dictionary analysis was performed to find the exact key.

Particle swarm optimization (Park *et al*., 2010; AlRashidi and El-Hawary 2009; Montes de Oca *et al*., 2009, M. S. Ramli *et al*., 2009) has demonstrated its usefulness as a optimization tool in recent years. The advantage of PSO is that this can be effectively used for multimodal problems and hence all suitable candidates satisfying the fitness function can be obtained. Thus the problem is reduced in finding the candidate solution which is one among the probable candidates. This was applied to the problem of Vigenère cipher and the results were found

Vigenère cipher: The Vigenère cipher proposed by Blaise de Vigenère from the court of Henry III of France in the sixteenth century is a progressive polyalphabetic substitution method. The set of related mono alphabetic substitution rules makes use of 26 Caesar Ciphers with shifts 0-25. The table used for encryption can be created for simple alphabet A to E which can be extended to all letters from A- Z is shown in Table 1. Each row in a table can be created by a simple shift of the previous row. Thus a Vigenère cipher of keyword length one can be considered as a Caesar cipher as this involves only one shift of the alphabets and thus forming a Caesar cipher.

To derive the cipher text using the Table, for each letter in the plain text, one finds the intersection of the row given by the corresponding keyword letter and the column given by the plaintext letter.

Corresponding Author: Ganapathi Sivagurunathan, Faculty of Computer Science and Engineering, Anna University, Coimbatore, Tamilnadu, India

Table 1: Vigenère table for alphabet A-E

Plaintext key	A	B	C	D	E
A	A	B	C	D	E
B	B	C	D	E	A
C	C	D	E	A	B
D	D	E	A	B	C
E	E	A	B	C	D

It can be modelled as C=(P +K) %26 where C is the cipher text and P,K are Plaintext and Key word letters respectively. Decipherment of an encrypted message is equally straightforward. This time one uses the keyword letter to pick a row of the Table and then traces down the row to the column containing the cipher text letter. The index of that column is the plain text letter. It can be modelled mathematically as, P = (C-K) % 26 .The main problem in breaking Vigenère ciphers is that the key length is unknown. Once the key length has been established, the cryptanalysis is reduced to analysing a number of Caesar ciphers, one for each character of the key. There are a few different approaches to finding the key length. The original method presented by Kasiski involves finding the distances between repeated patterns in the crypto text and factoring the most frequently occurring distances. Vigenère masks the frequency with which a character appears in a language: One letter in the cipher text corresponds to multiple letters in the plaintext and thus it makes the use of frequency analysis more difficult. It can be also seen that any message encrypted by a Vigenère cipher is a collection of as many Caesar shift ciphers as there are letters in the key.

Particle swarm optimisation: The goal of an optimization task is to find the parameters in the search space that maximizes the profit or minimizes the cost of a function. Particle Swarm Optimisation (PSO) is an optimization technique used to explore the search space of a given problem to find the value of the parameters involved in the function. This method described by Kennedy and Eberhart (1995) originates from the swarm intelligence of some animals and evolutionary computation. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbours (or colleagues). For a D-dimensional search space the position of the i^{th} particle is represented as $Xi = (xi_1, xi_2, \ldots, xi_D)$. Each particle maintains a memory of its previous best position $P_{besti} = (p_{i1}, p_{i2} \ldots p_{iD})$.

The best one among all the particles in the population is represented as $P_{gbest} = (p_{g1}, p_{g2} \ldots p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots v_{iD})$. In every iteration, P vector of the particle with best fitness in the local neighbourhood, designated P_g and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by: (Eq. 2)

$$v_{id}^{t+1} = wv_{id}^{t+1} + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{gd}^t) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (2)$$

The first part of Eq. 1 represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal thinking of the particle, the third part represents the cooperation among particles and is therefore named as the social component (Li, 2010; Kennedy *et al*., 2001). Acceleration constants c_1, c_2 (Kennedy, 1997; Eberhart and Shi, 2001) and inertia weight w (Shi and Eberhart, 1998) are the predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of [0, 1]. Particle's velocities on each dimension are clamped to a maximum velocity V_{max}. The velocity in a dimension is limited to V_{max} , if the sum of accelerations cause the velocity on that dimension to exceed V_{max}, which is 26 (as 26 alphabets in English Language). This process is repeated for a predetermined number of iterations or till a desired fitness value is reached.

MATERIALS AND METHODS

Plaintexts of various sizes up to 1Kb were taken for this experiment. These texts were taken mainly from different textbooks. Keywords of varying lengths up to thirty were used and the plain texts were converted into cipher texts using Vigenère method. To crack these cipher texts first we need to know the keyword length. These lengths were found using Coincidence test (Friedman, 1922; Ganesan and Sherman 1994). Once the keyword length is known then the problem is to find each character in the keyword. Frequency analysis (Friedman, 1980) is a useful tool which gives the average number of each alphabet normally present in an English text and it was employed. Fitness function based on the frequency of the occurrence of each alphabet was designed by using monogram and bigram statistics of English alphabets. Code was written using MATlab R2008 and was run on dual core personal computer.

Coincidence test: Since there are 26 letters in English alphabet, the probability of randomly choosing any given alphabet is 1/26 for all 26 alphabets. Similarly the probability of having the same alphabet twice is 1/26 * 1/26. If P(x) is the probability of 'x' occurring in any given plain text then P(a) = 0.32/400, P(b) = 0.6/400, P(c) = 0.12. Thus probability of 'a' followed by 'a' is 0.32/400 *0.32/400 , P(cc) = 0.12/400 * 0.12/400 and P(zz) = 0.1/400 * 0.1/400. The sum of the square of the probabilities of each alphabet gives the value of 0.0683 for plaintexts and 0.0385 for random texts. This gives us the coincidence count. These values can be used effectively to identify when two texts are likely to contain meaningful information in the same language using the same alphabet, to discover periods for repeating keys and to uncover many other kinds of nonrandom phenomena within or among cipher texts.

The same idea can be applied to a single text, where the sample is in effect compared with itself. Mathematically it can be computed the index of coincidence IC for a given letter-frequency distribution as: (Eq. 3)

$$IC = \frac{\sum_{i=1}^{c} n_i (n_i - 1)}{N(N-1)/c} \quad (3)$$

where, N is the length of the text and n_1 through n_c are the frequencies (as integers) of the c letters of the alphabet (c = 26 for English). The sum of the n_i is necessarily N.

The products n(n-1) count the number of combinations of n elements taken two at a time. Each of the n_i occurrences of the i-th letter matches each of the remaining n_i -1 occurrences of the same letter. There are a total of N(N-1) letter pairs in the entire text and 1 / c is the probability of a match for each pair, assuming a uniform random distribution of the characters. Thus, this formula gives the ratio of the total number of coincidences observed to the total number of coincidences that one would expect.

The expected average value for the I.C. can be computed from the relative letter frequencies f_i of the source language: (Eq. 4)

$$IC_{expected} = \frac{\sum_{i=1}^{c} f_i^1}{1/c} \quad (4)$$

If all c letters of an alphabet were equally distributed, the expected index would be 0.0683 for plaintext. So, this value can be calculated for every keyword length. The keyword length to which the given cipher text gives a coincidence value around 0.06 corresponds to the original keyword length. For example if the keyword length is 4 then for keyword lengths of 4 ,8 and 12 etc will have IC count of 0.06 . The maximum length can be safely assumed to be the keyword length.

Fitness function: To implement this fitness function, the frequency of each character in the decrypted text is calculated. This frequency is normalized by dividing it by the total number of characters in the file. This normalized frequency is then subtracted from the expected frequency of the character in normal English text. The absolute value of this difference is taken. The differences for all characters are added together. The normalization takes care that this value always lies between 0 and 1. The bigram is an extension of unigram to two characters. Now rather than calculating frequency of individual character, we calculate frequency of "pairs" of letters. For example, a pair "an" will always appear more frequently than pair "bt". Again statistics for the frequencies of these pairs are also available. These statistics are compared with the statistics obtained from the decrypted text. To implement this fitness function, the frequency of each pair of letters in the decrypted text is calculated. This frequency is normalized by dividing it by the total number of pairs in the file. This normalized frequency is then subtracted from the expected frequency of the pair in normal English text. The absolute value of this difference is taken. The differences for all pairs are added together. The normalization takes care that this value always lies between 0 and 1. The fitness function based on monogram and bigram is given by: (Eq. 5)

$$\text{fitness} = \sum_{i=1}^{zo} a * |SF(1) - DF(i)| + \sum_{i=1}^{zo}\sum_{j=2}^{zo} b * |SDF(1) - DDF(1)| \quad (5)$$

Here the letters A…Z are referenced by the indices 1…26, SF(i) is the standard frequency of character i in English, DF(i) is the measured frequency of the character i in English. SDF is the standard bigram frequency and DDF is the decoded bigram frequency. If the experimental key is closer to the key employed then this difference will be less and if this difference is large then the experimental key is not closer to the key employed. Now this problem has been reduced to an optimisation problem where it is required to reduce the error or the difference in the fitness function to minimum.

Application of PSO: The objective is to minimize the error in the objective function which is the difference of the expected frequency count with the observed frequency count of the decrypted cipher text. Since we deal with statistical values it is possible to get exact keyword only if the cipher texts are large enough (say > 50 Kb). With cipher texts which are less than 1Kb it is difficult to get the exact keyword or to put in other words, there may be multiple candidates satisfying the minimum fitness value and hence it is possible to have several alternate solutions. Mathematically we have a multimodal function which has several valleys. PSO is capable of solving multimodal objective functions and it was applied to this problem. Swarm size was selected as 100 and the number of iterations was limited to 25. Each character in the keyword is 26 dimensional since the character maybe any alphabet from A-Z.

PSO was applied and it was seen that the global minima in each character was intimated to other members in the swarm. The individual moved towards the global minima if necessary and found its local best solution. This process was repeated till the overall fitness of swarm reached a tolerance value or 100 iterations whichever was earlier.

The parameters of the optimization function were:

- Acceleration constants c_1,c_2 = 2.0
- Inertia weight w = 0.9
- Random weights r_1,r_2 = 0.4

RESULTS

Sample plain texts were taken from different text books and keywords of varying length 5-30 were applied and their corresponding cipher texts were obtained. Thus the program was tested for different cipher texts with different keyword lengths. The search space size is calculated based on brute force attack. The results are tabulated in Table (2 and 3).

Normally, the keyword length will be around ten as it may be difficult to remember bigger length keywords. So, experiments were conducted to determine the minimum required size of the cipher text so that the keywords can be found without error.

DISCUSSION

The solution space for each cipher text was performed with different weights to the digram frequency. The process was repeated with two different weights and the intersection of these spaces were selected and this provides the reduced search space.

Table 2: Reduction in search space for different keyword lengths

Keyword length	Size of search space	Alternate solutions obtained (average)	Reduction in search space (%)
5	11881376	2	100
10	1.411E+14	2	100
15	1.677E+21	6	100
20	1.992E+28	8	100
25	2.367E+35	8	100

Table 3: Minimum size of cipher text required for different keyword lengths

Keyword length	Size of cipher text required (minimum)
5	200 characters
10	400 characters
25	1024 characters (1 Kb)

From Table 2,It was seen that for smaller keyword lengths, the search space is so reduced that it provides one or two possible solutions, but as the keyword length increases then the possible solutions also increases , but by a small value. It can be seen that the maximum possible solutions obtained was only 8 and it was for keyword length of 25. Hence, the minimum number of cipher characters required for different keyword lengths was found and shown in Table 3. It can be safely stated that 1kb of cipher text is sufficient to cryptanalyse Vigenere cipher whose keyword length is less than or equal to 25.

When tested with keyword lengths greater than 25, it was found that one or two characters were erroneously identified. This was due to the fact that only 1024 cipher text characters were available and when they are distributed to 30 characters of keyword then for each character of the keyword we would get less than 35 characters (approx), which is not sufficient for applying frequency analysis. But it was seen that even with this lesser characters it was able to find most of the characters in the keyword. If the cipher text is >1Kb or if the keyword length is less than 30 then it is possible to find the keyword using reduced search space provided by the PSO with cipher text of size 1Kb.

It was also considered that a keyword length of 10 was normally the length of keyword employed. When it becomes large people tend to forget the keyword and hence tests were carried out with cipher texts of size 100 and 200 characters and with keyword lengths of 8 and 10. It was already established that for keyword lengths greater than 5 we need at least 400 characters of cipher text to find the keyword. With the limitation of available cipher text size reduced to 100 and 200 characters it was not able to find the whole keyword but a portion of the keyword was found as shown in Fig. 1 and 2.

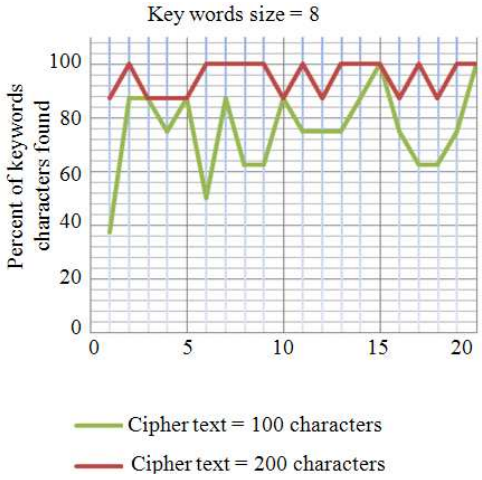

Fig. 1: Percentage of keyword found for keyword length of 8 with varying size of cipher text

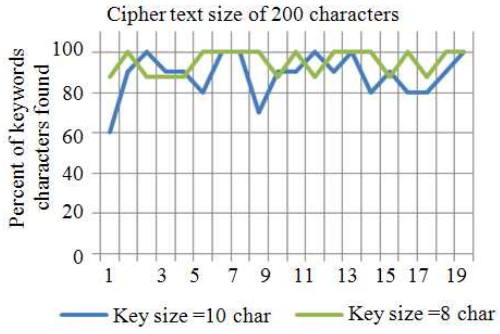

Fig. 2: Percentage of keyword found for fixed cipher text size with varying length of keyword

Minimum 30% of keyword was found when the cipher texts were of 100 characters and when the size of cipher texts were doubled the minimum keyword percentage found rose to 80%. It was also seen that for 60% of the given cipher texts, all the keyword characters were found, where as it was only 20% for 100 character cipher texts.

For a given cipher text of size 200 characters it was able to find minimum of 60% of keyword characters for a keyword length of 10 characters. But, when the keyword length got decreased to 8, nearly 90% of keyword characters were found. In both the cases, few 100% keyword characters were also found.

CONCLUSION

Experiments were conducted on Vigenère cipher with varying keyword and cipher text sizes. It was seen that the size of the reduced solution space using PSO is negligible as shown in Table 2. Further, from Table 3 we were able to calculate the minimum required size of cipher text required to find the keyword for different keyword sizes. It was also seen that even if the available cipher text is lesser than that the required size then more than 50% of the keyword can be found depending on the size of cipher text available. Thus it can be concluded that PSO performs better than Genetic algorithm with lesser amount of cipher text. The amount of cipher text required is only 25% of the cipher text required for genetic algorithm (Purusothaman *et al*., 2009). With 1Kb of available cipher text a keyword size of 25 was easily found.

REFERENCES

AlRashidi, M.R. and M.E. El-Hawary, 2009 A survey of particle swarm optimization applications in electric power systems. IEEE Trans. Evolut. Comput., 13: 913-918. DOI: 10.1109/TEVC.2006.880326

Al-Saidi, N.M.G. and M.R.M. Said, 2009. A new approach in cryptographic systems using fractal image coding. J. Math. Stat., 5: 183-189. **DOI:** 10.3844/jmssp.2009.183.189

Eberhart, R.C. and Y. Shi, 2001. Particle swarm optimization: developments, applications and resources. Proceeding of the IEEE International Conference on Evolutionary Computation, May 27-30, IEEE Xplore Press, Seoul, South Korea, pp: 81-86. DOI: 10.1109/CEC.2001.934374

Friedman, W.F., 1922. The Index and Coincidence and its Applications in Cryptography. 1st Edn., L. Fournier, Paris, pp: 87.

Friedman, W.F., 1980. Military Cryptanalysis. 1st Edn., Aegean Park Press, Laguna Hills, Calif, ISBN: 0894120441, pp: 77.

Ganesan, R. and A.T. Sherman, 1994. Statistical techniques for language recognition: An empirical study using real and simulated English. Crytologia, 8: 289-331. DOI: 10.1080/0161-119491882919

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceeding of the IEEE International Conference on Neural Networks, Nov. 27-1, IEEE Xplore Press, Perth, WA, Australia, pp: 1942-1948. DOI: 10.1109/ICNN.1995.488968

Kennedy, J., 1997. The particle swarm: Social adaptation of knowledge. Proceeding of the IEEE International Conference on Evolutionary Computation, Apr. 13-16, IEEE Xplore Press, Indianapolis, IN, USA, pp: 303-308. DOI: 10.1109/ICEC.1997.592326

Kennedy, J.F., J. Kennedy, R.C. Eberhart and Y. Shi, 2001. Swarm Intelligence. 1st Edn., Morgan Kaufmann, San Francisco, ISBN: 1558605959, pp: 512.

Li, X., 2010. Niching without niching parameters: Particle swarm optimization using a ring topology. IEEE Trans. Evolut. Comput., 14: 150-169. DOI: 10.1109/TEVC.2009.2026270

Montes de Oca, M.A., T. Stutzle, M. Birattari and M. Dorigo, 2009 Frankenstein?s PSO: A composite particle swarm optimization algorithm. IEEE Trans. Evolut. Comput., 13: 1120-1132. DOI: 10.1109/TEVC.2009.2021465

Park, J.B., Y.W. Jeong, J.R. Shin and K.Y. Lee, 2010 An improved particle swarm optimization for nonconvex economic dispatch problems. IEEE Trans. Power Syst., 25: 156-166. DOI: 10.1109/TPWRS.2009.2030293

Purusothaman, T., V. Gopalakrishnan, S. Arumugam, V. Palanisamy and S. Balraja *et al*., 2009. Cryptanalysis of vigenere cipher using genetic algorithm and dictionary analysis. The IASTED International Conference on Technology for Education.

Ramli, M.S., T.R. Ismail, M.A. Ahmad, S.M. Nawi and M.V. Mat, 2009. Improved coupled tank liquid levels system based on swarm adaptive tuning of hybrid proportional-integral neural network controller. Am. J. Eng. Applied Sci., 2: 669-675. DOI: 10.3844/ajeassp.2009.669.675

Shi, Y.H. and R.C. Eberhart, 1998. A modified particle swarm optimizer. Proceeding of the IEEE International Conference on Evolutionary Computation, May 4-9, IEEE Xplore Press, Anchorage, AK, USA., pp: 69-73. DOI: 10.1109/ICEC.1998.699146