Symmetric Cryptography based on Extended Genetic Algorithm

Anshu Arora[1], Himanshu Shekhar[2], Janahvi Shukla[3], Kumar Priybhash Singh[4]

[1,2,3,4]*Information Technology Department, Galgotias College of Engineering & Technology, Greater Noida, Uttar Pradesh – 201306*

[1]anshu9@live.com, [2]himanshushekhar59@gmail.com, [3]janhvi22shukla@gmail.com, [4]priybhash@live.com

Abstract. Genetic algorithms are a class of optimization algorithms. GAs attempt to solve problems through modelling a simplified version of genetic processes. There are many problems for which a GA approach is useful. It is, however, undetermined if cryptanalysis is such a problem. Therefore, work explores the use of GAs in cryptography. Both traditional cryptanalysis and GA-based methods are implemented in software. The results are then compared using the metrics of elapsed time and percentage of successful decryptions. A determination is made for each cipher under consideration as to the validity of the GA-based approaches found. In general, these GA-based approaches are typical of the field.

Keywords: Genetic algorithm, cryptography.

1. INTRODUCTION

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The application of a genetic algorithm (GA) to the field of cryptanalysis is rather unique. This nontraditional application is investigated to determine the benefits of applying a GA to a cryptanalytic problem. If the GA-based approach proves successful, it could lead to faster, more automated cryptanalysis techniques. However, since this area is so different from the application areas where GAs developed, the GA-based methods will likely prove to be less successful than the traditional methods. The primary goals of this work are to produce a performance comparison between traditional cryptanalysis methods and genetic algorithm based methods, and to determine the validity of typical GA-based methods in the field of cryptanalysis. The focus will be on classical ciphers, including substitution, permutation, transposition, knapsack and Vernam ciphers. The principles used in these ciphers form the foundation for many of the modern cryptosystems. Also, if a GA-based approach is unsuccessful on these simple systems, it is unlikely to be worthwhile to apply a GA-based approach to more complicated systems. A thorough search of the available literature found GA-based attacks on only these ciphers. In many cases, these ciphers are some of the simplest possible versions. For example, one of the knapsack systems attacked is the Merkle-Hellman knapsack scheme. In this scheme, a super increasing sequence of length n is used as a public key. According to Shamir [6] this is almost an order of magnitude below the typical case, and makes these attacks useful only as proof-of-concept examples for learning the use of GAs. In other GA-based attacks, there is a huge gap in the knowledge needed to re-implement the attack and the information provided in the paper. For example, the values of p and h are vitally necessary for an implementation of the Chor-Rivest knapsack scheme. This system utilizes a finite field of characteristic and the discrete logarithm problem is feasible. In the GA-based attack on this cipher, these values are not provided. This information hole means that this attack could have been run using trivial parameters, or parameters known to be easy to attack. This attack can not be considered remotely valid without knowing these parameters. Many of the GA-based attacks also lack information required for comparison to the traditional attacks. The number of generations it took for a run of the GA to get to some state is not at all useful when attempting to compare and contrast attacks. The most coherent and seemingly valid attacks were re-implemented so that a consistent, reasonable set of metrics could be collected. These metrics include elapsed time required for the attack and the success percentage of each attack. The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. The main idea is that in order for a population of individuals to adapt to some environment, it should behave like a natural system. This means that survival and reproduction of an individual is promoted by the elimination of useless or harmful traits and by rewarding useful behavior. The genetic algorithm belongs to the family of evolutionary algorithms, along with genetic programming, evolution strategies, and evolutionary programming. Evolutionary algorithms can be considered as a broad class of stochastic optimization techniques. An evolutionary algorithm maintains a population of candidate solutions for the problem at hand. The population is then evolved by the iterative application of a set of stochastic operators. The set of operators usually consists of mutation, recombination, and selection or something very similar. Globally satisfactory, if sub-optimal, solutions to the problem are found in much the same way as populations in nature adapt to their surrounding environment. Using Tomassini's terms [8] genetic algorithms (GAs) consider an optimization problem as the environment where feasible solutions are the individuals living in that environment. The degree of adaptation of an individual to its environment is the counterpart of the fitness function evaluated on a solution. Similarly, a set of feasible solutions takes the place of a population of organisms. An individual is a string of binary digits or some other set of symbols drawn from a finite set. Each encoded individual in the population may be viewed as a representation of a particular solution to a problem. In general, a genetic algorithm begins with a randomly generated set of individuals. Once the initial population has been created, the genetic algorithm enters a loop [8]. At the end of each iteration, a new population has

ISBN: 9788175157538

been produced by applying a certain number of stochastic operators to the previous population. Each such iteration is known as a generation [8]. A selection operator is applied first. This creates an intermediate population of n "parent" individuals. To produce these "parents", n independent extractions of an individual from the old population are performed [8].The probability of each individual being extracted should be (linearly) proportional to the fitness of that individual. This means that above average individuals should have more copies in the new population, while below average individuals should have few to no copies present, i.e., a below average individual risks extinction. Once the intermediate population of "parents" (those individuals selected for reproduction) has been produced, the individuals for the next generation will be created through the application of a number of reproduction operators [8]. These operators can involve one or more parents. An operator that involves just one parent, simulating asexual reproduction, is called a mutation operator. When more than one parent is involved, sexual reproduction is simulated, and the operator is called recombination [8].The genetic algorithm uses two reproduction operators-crossover and mutation. To apply a crossover operator, parents are paired together. There are several different types of crossover operators, and the types available depend on what representation is used for the individuals. For binary string individual's one-point, two-point and uniform crossover is often used. For permutation or order-based individuals, order, partially mapped, and cycle crossover are options. The one-point crossover means that the parent individuals exchange a random prefix when creating the child individuals. Two-point crossover is an exchange of a random substring and uniform crossover takes each bit in the child arbitrarily from either parent. Order and partially mapped crossover are similar to two-point crossover in that two cut points are selected. For order crossover, the section between the first and second cut points is copied from the first parent to the child [10]. The remaining places are filled using elements not occurring in this section, in the order that they occur in the second parent starting from the second cut point and wrapping around as needed. For partially mapped crossover, the section between the two cut points defines a series of swapping operations to be performed on the second parent [10]. Cycle crossover satisfies two conditions - every position of the child must retain a value found in the corresponding position of a parent, and the child must be a valid permutation. Each cycle, a random parent is selected. The successful attacks were those on the transposition and permutation ciphers by Matthews Clark. These attacks were further investigated in an attempt to improve or extend their success. Unfortunately, this attempt was unsuccessful, as was the attempt to apply the Clark [11] attack to the mono alphabetic substitution cipher and achieve the same or indeed any level of success. Overall, the standard fitness equation genetic algorithm approach, and the scoreboard variant thereof, are not worth the extra effort involved. Traditional cryptanalysis methods are more successful, and easier to implement. While a traditional method takes more time a faster unsuccessful attack is worthless. The failure of the genetic algorithm approach indicates that supplementary research into traditional cryptanalysis methods may be more useful and valuable than additional modification of GA-based approaches.

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations [3]. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm [18]. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached [1]. Genetic algorithms find application in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields [1].

A typical genetic algorithm requires:

1. A genetic representation of the solution domain,
2. A fitness function to evaluate the solution domain.

The basic generic algorithm cycle

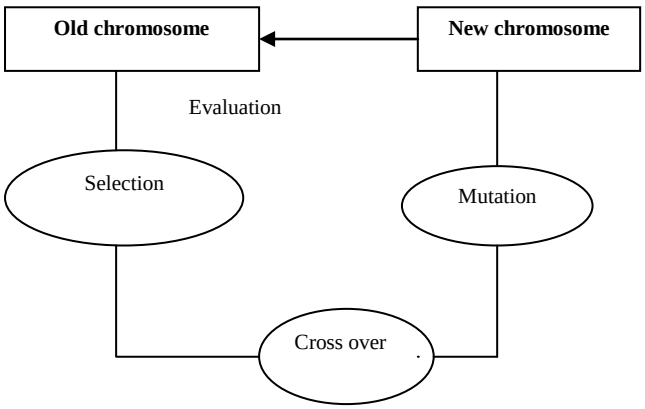
A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations [2]. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming. The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent [2]. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise [4].

2. METHODOLOGY

1) Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions [19]. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

ISBN: 9788175157538



During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected [7]. Certain selection methods rate the fitness of each solution and preferentially select the best solutions [7]. Other methods rate only a random sample of the population, as this process may be very time-consuming.

2) Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents" [5]. New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests more than two "parents" are better to be used to reproduce a good quality chromosome. These processes ultimately result in the next generation population of chromosomes that is different from the initial generation [15].

3. RELATED WORKS

In general, the genetic algorithm approach has only been used to attack fairly simple ciphers. Most of the ciphers attacked are considered to be classical, i.e. those created before 1950 A.D. which have become well known over time [9]. Ciphers attacked include:

- Monoalphabetic Substitution cipher
- Polyalphabetic Substitution cipher
- Permutation cipher
- Transposition cipher
- Merkle-Hellman Knapsack cipher
- Chor-Rivest Knapsack cipher
- Vernam cipher

Polyalphabetic Substitution cipher

This cipher is a more complex version of the substitution cipher. Instead of mapping the plaintext alphabet onto cipher text characters, different substitution mappings are used on different portions of the plain text. The result is called poly alphabetic substitution. The simplest case consists of using different alphabets sequentially and repeatedly, so that the position of each plaintext character determines which mapping is applied to it [14]. Under different alphabets, the same plaintext character is encrypted to different cipher text characters, making frequency analysis harder.

This cipher has the following properties (assuming m mappings):

- The key space L consists of all ordered sets of m permutations
- These permutations are represented as $(l_1, l_2, \ldots, l_m)$
- Each permutation l_i is defined on the alphabet in use
- Encryption of the message $x = (x_1, x_2, \ldots, x_m)$ is given by

$$e_k(x) = l_1(x_1)l_2(x_2) \ldots l_m(x_m),$$

assuming the key $L = (l_1, l_2, \ldots, l_m)$

The encryption and decryption functions could also be written as follows (assuming that the addition/subtraction operations occur modulo the size of the alphabet):

- $e_L(x_1, x_2, \ldots, x_m) = (x_1 + l_1, x_2 + l_2, \ldots, x_m + l_m)$
- $d_L(y_1, y_2, \ldots, y_m) = (y_1 - l_1, y_2 - l_2, \ldots, y_m - l_m)$

The most common poly alphabetic substitution cipher is the Viennese cipher. This cipher encrypts m characters at a time, making each plaintext character equivalent to m alphabetic characters. A example of the Vigenere cipher is:

- Suppose m = 6
- Using the mapping A $ 0, B $ 1,……., Z $ 25
- If the keyword K = CIPHER, numerically (2, 8, 15, 7, 4, 17)
- Suppose the plaintext is the string ‘cryptosystem’
- Convert the plaintext elements to residues modulo 26 - 2 17 24 15 19 14 18 24 18 19
- Write the plaintext elements in groups of 6 and add the keyword modulo 26

  2 17 24 15 19 14 18 24 18 19 4 12
  2 8 15 7 4 17 2 8 15 7 4 17
  4 25 13 22 23 5 20 6 7 0 8 3
- The alphabetic equivalent of the cipher text string is: EZNWXFUGHAID.
- To decrypt, the same keyword is used but is subtracted modulo 26 from the cipher text

Vernam cipher

The Vernam cipher is a stream cipher defined on the alphabet A = {0, 1}. A binary message $m_1m_2\ldots m_t$ is operated on by a binary key string $k_1k_2\ldots k_t$ of the same length to produce a cipher text string $c_1c_2\ldots c_t$ where $c_i=m_i-k_i$, 1=i-t [21]. If the key string is randomly chosen and never used again, this cipher is called a one-time system or one-time pad. The one-time pad can be shown to be theoretically unbreakable [21]. If a cryptanalyst has a cipher text string $c_1c_2\ldots c_t$ encrypted using a random, non-reused key string, the cryptanalyst can do no better than guess at the plaintext being any binary string of length t. It has been proven that an unbreakable Vernam system requires a random key of the same length as the message [21].

ISBN: 9788175157538

4. CONCLUSION

The primary goals of this paper is to produce a cryptanalysis methods and genetic algorithm (GA) based methods, and to determine the validity of typical GA-based methods in the field of cryptanalysis. The focus was on classical ciphers, including substitution, permutation, transposition, knapsack and Vernam ciphers. The principles used in these ciphers form the foundation for many of the modern cryptosystems. Also, if a GA-based approach is unsuccessful on these simple systems, it is unlikely to be worthwhile to apply a GA-based approach to more complicated systems. A thorough search of the available literature resulted in GA based attacks on only these ciphers [17]. In many cases, these ciphers were among the simplest possible versions. Many of the GA-based attacks lacked information required for comparison to the traditional attacks. Dependence on parameters unique to one GA-based attack does not allow for effective comparison among the studied approaches [16]. The most coherent and seemingly valid GA-based attacks were re-implemented so that a consistent, reasonable set of metrics could be collected [17]. The main metric for both classical and genetic algorithm attacks was elapsed time. In the classical algorithm case, the elapsed time was comprised almost completely of time spent interacting with the user, while in the genetic algorithm case, it was comprised almost entirely of processing time. This difference in composition of elapsed time makes it difficult to compare the attacks on a time basis. Therefore, an additional metric was required. The secondary metric used was the percentage of successful attacks per test set. Success was defined as complete decryption of the cipher text. This metric was only calculated for the GA-based attacks, as a classical attack will run until the text is decrypted [20]. A GA-based attack, on the other hand, runs for a specific number of generations, independent of the decryption of the cipher text. Using this metric, as well as elapsed time, gives one a better feel for the usefulness of each attack. The time measurement is irrelevant if the attack is unsuccessful. These two metrics were chosen so that traditional and GA-based attacks could be compared [20]. Elapsed time measures the efficiency of the attack while the percentage of successful attacks measures how useful the attack is. Both of these metrics are needed to gain a complete picture of an attack. Of the twelve genetic algorithm based attacks found in the literature, seven were selected for re-implementation. None of the knapsack or Vernam cipher attacks were reimplementation, for varying reasons. Of these seven, only three attacks were successful in decrypting any of the cipher texts. The successful attacks were those GA-based attacks on the permutation and transposition ciphers. Clark's [13] GA-based attack on the permutation cipher was by far the most successful. Both of the attacks which used a scoreboard for fitness calculation, Matthews [12] and Clark [13], achieved at least some success. The third successful GA-based attack was by Grndlingh and Van Vuuren [13], the only successful attack to use a fitness equation. These three successful attacks were extended to further examine their success rates. The two transposition attacks were tested using much larger population sizes and numbers of generations, while the permutation attack was attempted on larger block lengths. The two transposition cipher attacks obtained about the same success rate as before, while the permutation cipher attack experienced decreased success. Since the scoreboard approach was successful in both cases it was used, this style of GA-based attack was attempted on the mono alphabetic substitution cipher. The attack was based on Clark's attack[13], simply altered to apply to a different cipher. Unfortunately, this attempt was unsuccessful. No cipher texts were completely decrypted with either variant attempted.

5. FUTURE WORK

There are several areas in which this work could be expanded. One of these is in the investigation of the scoreboard approach to fitness calculation. This method achieved the best results of the two methods used, yet does not seem to be in widespread use. Applying this approach to other problems may prove useful. A comparison of the scoreboard approach to the fitness equation approach in other applications would be useful as well. This would determine whether the scoreboard approach is limited to specific applications only, or if it has widespread applicability. Another area for investigation is the use of less common fitness calculation measures. The scoreboard approach, for example, proved useful in this application despite the rarity of its use.

REFERENCES

[1] Alcott, Louisa May. Little Women Parts I & II. Great Literature Online 1997-2003. Retrieved August

[2] The Holy Bible, King James Version. New York: American Bible Society: 1999 Bartleby.com, 2000.

[3] Bryan, William Jennings, ed. The World's Famous Orations. New York: Funk and Wagnalls, 1906 New York: Bartleby.com, 2003. Retrieved August 20, 2003

[4] Kahn, D. (1967). The Codebreakers: The Story of Secret Writing. New York: Macmillan.

[5] Clark, A., & Dawson, Ed. (1997). A Parallel Genetic Algorithm for Cryptanalysis of the Polyalphabetic Substitution Cipher. Cryptologia.

[6] Shamir, A. (1982). A polynomial time algorithm for breaking the basic Merkle- Hellman cryptosystem. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 1982.

[7] Clark, A., & Dawson, Ed. (1998). Optimisation Heuristics for the Automated Cryptanalysis of Classical Ciphers. Journal of Combinatorial Mathematics and Combinatorial Computing.

[8] Tomassini, M. (1999). Parallel and Distributed Evolutionary Algorithms: A Review. In K. Miettinen, M. Makela, P. Neittaanmaki and J. Periaux (Eds.), Evolutionary Algorithm in Engineering and Computer Science . Chichester: J. Wiley

[9] Clark, A., Dawson, Ed, & Bergen, H. (1996). Combinatorial Optimisation and the Knapsack Cipher.

[10] Oliver, I.M., Smith, D.J., & Holland, J.R.C. (1987, July). A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In John J. Grefenstette (Ed.), Proceedings of the 2nd International Conference on Genetic

ISBN: 9788175157538

Algorithms, Cambridge, MA, USA, July 1987, (pp. 224-230). Lawrence Erlbaum Associates. and Sons.

[11] Clark, A. (1994). Modern optimisation algorithms for cryptanalysis. In Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, November 29- December 2.

[12] Matthews, R.A.J. (1993, April). The use of genetic algorithms in cryptanalysis. Cryptologia.

[13] Clark, A., Dawson, Ed, & Nieuwland, H. (1996). Cryptanalysis of Polyalphabetic Substitution Ciphers Using a Parallel Genetic Algorithm. In Proceedings of IEEE International Symposium on Information and its Applications, September 17-20.

[14] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). Introduction to Algorithms: Second Edition. Cambridge, Boston: MIT Press, McGraw-Hill.

[15] Darwin, Charles Robert. The Origin of Species. Vol. XI. The Harvard Classics. New York: P.F. Collier & Son..

[16] Gaines, H. F. (1956). Cryptanalysis: a Study of Ciphers and their Solutions. New York: Dover.

[17] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Boston: Addison-Wesley.

[18] Grundlingh, W. & van Vuuren, J. H. (submitted 2002). Using Genetic Algorithms to Break a Simple Cryptographic Cipher. Retrieved March 31, 2003.

[19] Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press.

[20] Kreher, D. L. & Stinson, D. R. (1999). Combinatorial Algorithms: Generation, Enumeration, and Search. Boca Raton: CRC Press.

[21] Menezes, A., van Oorschot, P., & Vanstone, S. (1997). Handbook of Applied Cryptography. Boca Raton: CRC Press.

ISBN: 9788175157538