Cryptanalyzing the Playfair Cipher Using Evolutionary Algorithms

Benjamin Rhew

December 9, 2003

Contents

Abstract:

Cryptanalysts need ways to automate the process of analyzing ciphers, so that cryptographic systems can be tested more efficiently. Evolutionary algorithms may be one way of automating the process. Writing an evolutionary algorithm to cryptanalyze the Playfair Cipher is the first step towards understanding how cryptanalysis and evolutionary algorithms might work together. However, the results obtained from the current project are not good, though with more work it should improve.

Keywords:

Evolutionary Algorithms, Cryptanalysis, Playfair Cipher.

1 Introduction

In today's world, cryptology as a form of computer security is becoming more and more important to computer users. Mostly this is through the use of cryptographic algorithms, which are used to encrypt messages and data to keep them safe. But there is another side to the field of cryptology: cryptanalysis. Most people do not know the term, but they understand the basic idea in a negative stereotypical fashion - "hackers" who try to break into protected data to steal its contents. However, cryptanalysis has its good points as well. Without it, algorithms couldn't be tested to see how secure they are. This is a plus for general computer users, as it helps more secure cryptographic algorithms to be developed and tested.

So, what is the connection between cryptanalysis and evolutionary algorithms? For starters, the cryptanalysis of an encrypted message is usually a time-consuming process. This is due to the large number of possible keys for most cryptographic algorithms. Because of the time involved, more and more research is going into finding ways to automate the process on the more difficult algorithms. To understand the idea, the methods are implemented on cryptographic systems that are easier to break, such as simple transposition and substitution ciphers, before proceeding onto more difficult cryptographic algorithms. This has been done with more conventional algorithms, but only preliminary work has been done with evolutionary algorithms.

In essence, the Playfair Cipher is another simple cipher to decrypt, though it is more difficult than the general transposition or subsitution ciphers. This algorithm is used because it is necessary understand the basics of using evolutionary algorithms to try to break cryptographic systems, and it is good to use a simple algorithm for testing purposes. It is hoped that the knowledge gained from this project will lead to being able to implement more difficult algorithms using evolutionary algorithms.

The rest of the paper is organized as follows. Section2 will explain the basics of cryptanalysis. Section 3 will cover the basics of the Playfair Cipher. The design considerations in implementing the algorithm will be explained in Section 4, while Section 5 will cover the experimentation and results obtained. Finally, the conclusions to my findings will be presented in Section 6.

2 The Basics of Cryptanalysis

To better understand some of the basic concepts behind this project, we first take a look at cryptanalysis. According to [**?**], cryptanalysis is "The branch of cryptology dealing with the breaking of a cipher to recover information, or forging encrypted information that will be accepted as authentic." For this discussion, we will be concerned with the first part of the definition, breaking ciphers.

Cryptanalysts have three basic types of attacks used while trying to break a cipher, all dependent on the type of information he or she has ([**?**]:

- Ciphertext only: With this method, the cryptanalyst knows the type of encryption algorithm used, and has a block of ciphertext to work with. Finding the cryptographic key is the most difficult with this method, but much of the time this is the only method that can be used.

- Known plaintext: This is somewhat similar to the ciphertext only attack, but the cryptanalyst has the added benefit of knowing the plaintext as well. This extra information gives the cryptanalyst a two-pronged attack on the algorithm, and it also helps as a form of verification when the cipher is finally broken.

- Chosen ciphertext: Sometimes the cryptanalyst is in a position where he or she can send a block of ciphertext to the user and receive the plaintext back. This benefits the cryptanalyst because he or she essentially has a large source of plaintext and ciphertext, which makes breaking the cipher much easier.

Besides these three basic attacks, there are a couple of variations on the chosen ciphertext attack that are less common - chosen plaintext and chosen text. Chosen plaintext is just like chosen ciphertext, except for the cryptanalyst sending plaintext and receiving ciphertext back. Chosen text, on the other hand, combines the chosen ciphertext attack with the known plaintext attack. These variations are less likely to be of use, since it requires the cryptanalyst to have more information than he or she is likely to get.

Once the cryptanalyst has chosen the type of attack to be used, he or she has to use knowledge of the algorithm to try and obtain the key. In a worst case scenario, the brute force technique will have to be used. This is where the cryptanalyst has to randomly run through the entire keyspace of the algorithm, testing each key on the ciphertext. Usually, however, the cryptanalyst can apply some domain knowledge to make the search space smaller.

However, some cryptographic systems are inherently weak, making them easy to break. The simpler cryptographic systems, including transposition and substituion ciphers, as well as the Playfair Cipher, are such systems. They have weaknesses that are very easy to exploit, such as:

- A comparatively small keyspace: This makes a brute force analysis of the keyspace possible in hours, even minutes, with the computers in use today.

- Language analysis: Since the simplest cryptographic systems only change letters around, the frequency of the letters used is the same as in the original plaintext, but shifted. Since some letters occur more frequently than others, an analyst can examine the frequency of the letters in the ciphertext to make good guesses towards the key. Slightly more advanced algorithms, such as the Playfair Cipher, make this more difficult by switching pairs of letters, but they are still relatively simple to analyze with this method.

Cryptanalysts can easily break these systems, but the best algorithms today are designed to be computationally infeasible to break. This means that the

Figure 1: The Playfair Cipher

keyspaces for the algorithms are so large that it would take hundreds or thousands of years to break, even using today's supercomputers. Even so, cryptanalysts still try to find new ways to attack the systems, so that they can find the weaknesses before the "hackers" do.

3 The Playfair Cipher

As mentioned earlier, the Playfair Cipher is one of the simpler ciphers that can be used to encrypt a message. It is slightly more complex than transposition and substitution ciphers, but it is still simple enough that it makes a good algorithm to use for trying out new ideas for cryptanalysis.

The mechanisms for the cipher are based on a 25 character key, arranged in a five by five grid (figure[1]). The letters I and J share a position in the key, which is why there are 25 letters instead of 26. The key is usually generated by picking some keyword with no repeating letters, and using it as the first row or two in the grid. The key can be generated randomly as well, though with the comparatively small search space it does not help much.

3.1 Encryption

Before encryption can take place, the message to be encrypted needs to be converted to the proper format. As an example, the following message will be converted:

This message will self-destruct in five seconds.

First, all spaces and non-alphabetic characters must be taken out of the message.

Thismessagewillselfdestructinfiveseconds

Now the message must be broken into pairs of 2 letters each, as well as converted to all uppercase. During this phase, if a pair consists of a single letter, such as ‘ss’, it must be broken up with a different letter, such as an ‘X’ or ‘Z’. This affects other possible pairings further along in the message, so it has to be converted from the start of the message on. Also, if the message ends with an unpaired letter, a filler letter needs to be added.

> TH IS ME SX SA GE WI LX LS EL FD ES TR UC TI NF IV ES EC ON DS

Once the message is in the proper format, it is relatively easy to encrypt it into ciphertext. There are three rules:

- If the pair of letters to be encrypted occupies the same row, then replace them with the pair of letters to the right, with wraparound. For example, using figure[1], TH becomes CI (or CJ).
- If the pair of letters occupies the same column, replace them with the pair of letters directly below them, with wraparoud. For example, SA becomes FS (figure[1]).
- If neither of the above is true, then replace the pair with the letter in the same row as the current letter, but in the column of the other letter. For example, ME becomes NB using figure[1].

Following these rules, the message is encrypted, with the resulting ciphertext as follows:

> CJ ST NB JY FS BF XH PW QH DP GE FJ GT KT CJ AE CX FJ BJ NA FH

3.2 Decryption

Decryption is very similar to encryption, with only some minor changes to the rules.

- If the pair of letters to be decrypted occupies the same row, instead of replacing them with the letters to the left, replace them with the pair of letters to the right, with wraparound. Using figure[1], ST becomes IS (or JS).
- Similarly, if the pair of letters occupies the same column, replace them with the pair of letters directly above them, with wraparound. For example, GT becomes TR (figure[1]).
- And if neither of the above are true, the last rule is the same as the one for encryption: replace the pair with the letter in the same row as the current letter, but in the column of the other letter.

Obviously, if the ciphertext generated by the encryption algorithm were decrypted, the plaintext would be recovered.

Figure 2: An Individual for the Evolutionary Algorithm

MONARCHISTBDEFGKLPQUVWXYZ

4 Implementing the Algorithm

Since the problem solution involves using an evolutionary algorithm to try and evolve better keys, there are several design considerations that need to be taken into account:

- What information are the individuals going to contain?
- How will the individuals be represented?
- How will parents be chosen?
- How will reproduction be implemented?
- How will the mutation operator mutate individuals?
- How will "survival of the fittest" be implemented?
- What will determine the fitness of each individual?

These are most of the major design factors that need to be considered while writing the implementation, and the implementations used will be explained in detail below.

First, let's consider the individuals. Each individual contains one possible key for the Playfair Cipher, and associated with each individual is it's fitness, though that is contained in a separate variable. The individuals are represented by an array of 25 characters, which are in turn held in another array to represent the total population(figure 2). A character array is the most natural method of representation in this case, since the keys are based on permutations of the alphabet. Next up is parent selection. Parent selection in this case is rank-based, wince it will be based on the fitness. This means that individuals with higher fitnesses receive a better chance to be chosen for reproduction. In a sense it isn't strictly rank-based, since the individuals are in random order, and those with lower fitness may be looked at before individuals with high fitness.

Next on the list is implementing reproduction. Each child will have two parents, but the parents can have one or more children. The child production process begins by taking one parent and randomly selecting characters to keep in the same position in the child. The other parent is used to fill in the rest of the child, taking each character not already in the child in order (figure 3). Following creation, the child will be randomly mutated by swapping any two of the characters in the individual (figure[3]).

The next generation of the population is determined based on the fitness of the individuals. The worst N individuals are chosen to die, where N is the

Figure 3: Reproduction in the Evolutionary Algorithm

MONARCHIST
CNHOSTIAMR
- O - - RC - IS -
NOHTRCAISM

Figure 4: Mutation in the Evolutionary Algorithm

NOHTRCAISM
NOMTRCAISH

total number of children. In other words, any individual whose fitness is low is eventually exterminated.

Determining the fitness of each individual is the most difficult part of the implementation. Since the objective is to find the key to the cipher, each possible key has to be used in the decryption algorithm. From there, the recovered plaintext must be compared to a dictionary of possible words, and a fitness returned based on the number of words found. This is not the simplest to implement, for several reasons. One is that words that have double letters may not be counted correctly, due to the fact that the double letter might be split up. Second, because I and J share a position in the key, all the words that have Is and Js in them have to be checked using both letters, if the dictionary is fully implemented. Third, the plaintext has no whitespace to delimit words, so being able to tell where words end and begin is difficult.

5 Experimentation and Results

After the implementation is completed, it is time to do some testing of the algorithm. Due to the complexity of the fitness function, it was expected that it would not run fast, and this turned out to be the case.

The test parameters for the experiment were an initial population of 100, with 50 parents and two children per set of parents. Unfortunately, it was during this first set of tests that a major problem was discovered in the implementation, causing individuals to be incorrect. After the first reproductive cycle, individuals would appear that had multiple instances of a single letter in them, or they would have missing letters, and so forth. This made trying to run the program for an extended period of time impossible, with results that made no sense. The bug causing this problem was never found.

Even with the problems, enough information was gathered to tell that the current implementation is not good. For instance, even with the simple version of the dictionary look-up, the fitness function takes three minutes to run for 100 individuals. That means that to have a chance of discovering the key for the cipher, the program generated would need to be run for several hours. Good cryptanalysts could discover the key by hand in under an hour, most likely, so this is unacceptably long. Hopefully this problem can be eliminated in future generations of the program.

6 Conclusion

The results of the experiment were not as good as hoped for, since the implementation does not work right in the first place. Even if the algorithm worked completely, the wait for it to discover the right key would be too long to be of any use.

As it stands, there is too little information learned at this point to be able to write an evolutionary algorithm to break more difficult ciphers. More work needs to be done on the current algorithm to understand it better, and maybe some studying of simpler algorithms is also in order.

In conclusion, this paper covered some of the basics of cryptanalysis, as well as the mechanics of the Playfair Cipher. The implementation of the evolutionary algorithm was explored in detail, and the experimental results obtained were covered.

References

[Sta03] William Stallings, 2003.

[Wik] Wikipedia. Cryptanalysis definition.