Encrypting Chaos:
Fractal Encryption

New Mexico Adventures in
Supercomputing Challenge
Final Report
April 2, 2003

Team Members
Nick Whitehead
Michael Overton
Zach LaBry
Franklin Hamilton
Brian Leising
Teacher
Jim Mims

I. Executive Summary

With the growing ease of electronic information transfer, comes the need to protect that data, and ensure its integrity. Such information can come from a variety of sources: personal, corporate, or governmental. Regardless of the particular contents of the information, a lot of proprietary or sensitive information exists, and must be stored on devices that are not safe from intruders. The science of cryptography deals with the problems of information security as it pertains to storage and transfer.

Most modern encryption algorithms are based on one of the following two categories of processes: mathematical problems which are easy and have inverses which are believed (but not proven) to be hard, and sequences of permutations that are defined in-part by the inputs given to them. The first category, which summarizes most public-key encryption methods suffers from the inability to prove the difficulty of the algorithms. The second method, which categorizes most ciphers, often suffers from theoretical correlations between the input ("plaintext") and the output ("ciphertext").

Fractals and chaotic systems have properties which have been extensively studied over the years, and derive their inherent complexity from the extreme sensitivity of the system to the initial conditions. Such systems have the property that no closed form solutions exist for them, and therefore "simple" formulas that exactly define the system at any given time do not exist. As applied to cryptography, this qualifies as a very hard problem. The major advantage that chaotic systems, such as the *n-body problem* used in this project, are *provably* hard, eliminating one of the fundamental drawbacks to conventional encryption.

II. Software Engineering Goal

The goal of this project is to design and develop an algorithm capable of securely encrypting data by using the properties of fractals. Cryptography can make use of the repeatability of fractals. Additionally, cryptography can make use of contrast between regions of a fractal. That is, since fractals can change value very dramatically over a short interval, an attacker trying to break a system using fractal cryptography would have difficulty telling when they were "close" to finding the solution to the problem.

III. Research

A. Cryptography

Cryptography is the science concerned with the transfer of information. Some of the basic goals of cryptography include information security, information integrity, authentication, and non-repudiation, among others. This project is concerned primarily with the first goal,

Most current methods of encryption come in two flavors: symmetric, and asymmetric. Symmetric cryptography deals with permutations of a fixed set of data based upon some key. The key for encryption, and the key for decryption in this case are the same key. Symmetric algorithms also come in two major classes: block ciphers, and stream ciphers. Block ciphers treat each block of data in the same manner, while stream ciphers will treat the same blocks differently depending upon what preceded them. The ultimate goal of a cipher is to achieve a level of security that one finds in a one-time pad.

A one-time pad is a random (truly random) sequence the same length as the message added modulo 2 to the message. The statistical distribution of bits of the

plaintext and ciphertext have absolutely no correlation to each other, and there is no repetition of the key sequence from which patterns can be found, so any message the same length of the original plaintext (and ciphertext) has an equal probability of being the correct message. The result is that an attacker would have no way of determining the correct message without having some additional information about it.

Ciphers (notable among them are DES, and AES/Rijndael) are unable to achieve this "perfect" level of security, however, because their key size is much smaller for practical reasons, and truly random sources are very difficult to obtain or make use of. Although the specific sequences of permutations used by these ciphers are rather difficult to attack, there are methods (linear and differential cryptanalysis, for example) which are able to find and exploit some weaknesses. Many of these are known-plaintext or chosen-plaintext (meaning the attacker was able to get one party to encrypt a specific piece of information), and are practically infeasible to exploit, but nevertheless, bring down the overall security of the algorithm from the ideal of a one-time pad.

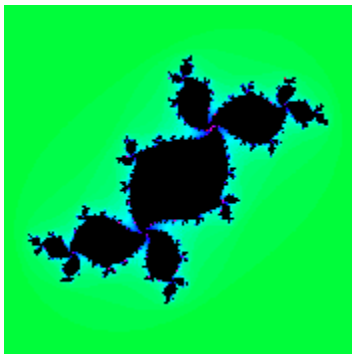
Asymmetric algorithms work in a very different way. In these algorithms, one key is known to the public, and is used to encrypt information to send to a certain receiver with the corresponding private key. The private and public keys are both different, removing the need for key exchange (or making exchange of symmetric keys safe, as is usually the case).

Most of these public-key encryption algorithms rely on problems which are NP-Complete, and thought to be very hard. Solutions to NP-Complete problems can be checked in polynomial time. It is also thought that it is impossible to solve these problems in polynomial time, although recent literature leans toward the contrary,

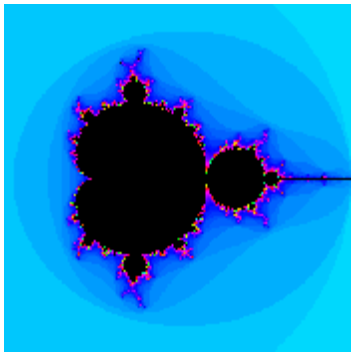
meaning that the security of these algorithms could potentially be compromised [Bolotashvili]. The specific problems which are addressed by many algorithms are the discrete logarithm problem (which asserts that it is easy to exponentiate something, but difficult to find the log), and the theory of elliptic curves.

B. Fractals

Fractals are seen everywhere in nature and yet so very mysterious. First imagined by Julia and Mandelbrot, fractals have an element of chaos, but also have an essence of order. Look at the pictures blow. They both have a random nature, but also have an element of order. But these are not the only way to represent fractals. They are ultimately based in mathematics.



- Julia Set



- Mandelbrot Set

Take, for example, the following function. $F(x)= x^2 + C$. Set C as some constant, then give an initial condition for the value of x. Calculate the value of the function, then take the value just obtained and plug it in as your new x-value. This is an operation called

iteration, and is the core concept of fractals. While the previous is a very simple example, the one that we are using is much more complex. We are using the iterative nature of the N-body problem to create our algorithm.

C. Mathematical Model: The N-Body Problem

Any particle that has mass obeys Newton's law of gravitation (on the macroscopic scale, at least). This law sates that the force of attraction between two bodies is proportional to the product of their masses, and inversely proportional to the square of the distance between their centers, giving:

$$F = \frac{-Gm_1 m_2}{r^2},$$

where G is the constant of proportionality (in this case, called the universal constant of gravitation).

When looking at a particular object, the signed magnitude of its acceleration vector can be determined by dividing the mass, which leaves this quantity as a function of the mass of the other object, and the distance between them. Acceleration, however, is the second derivative of an object's position with respect to time, so the resultant equation is:

$$\frac{d^2 P}{dt^2} = \frac{-Gm}{r^2},$$

where P is the position of the object as a function of time.

We can now see that in a space of n-dimensions, based upon the Euclidean distance formula, each coordinate of the particle's acceleration can be shown to have the form:

$$\frac{d^2 x_k}{dt^2} = \frac{-Gm}{\sum_{i=1}^{n} (x_i - y_i)^2},$$

where the x values are the coordinates of the object being observed, and the y values are the coordinates of the object toward which it is moving.

Second order differential equations of this form can be solved in closed form, but as more objects are added that influence the one we are observing, the situation becomes much more complex, and it is no longer possible to exactly solve the equation. The resultant equation is of the form:

$$\frac{d^2 x_k}{dt^2} = \sum_{j=1}^{p} \left(\frac{-Gm_p}{\sum_{i=1}^{n} \left(x_i - \left(y_p \right)_i \right)^2} \right),$$

where p is the number of objects in the system acting upon the one we are observing.

The n-body problem is concerned with finding determining the motion of particles while acting under the influence of radial force fields projected from point sources (as such, it is classified as a dynamic system). These forces take the form of the one discussed above, but differ simply in the constant that is used in the equation. The position of a particle under such conditions is defined by a system of differential equations - namely, Newton's laws of motion. For the specialized case of the 2-body problem (and the trivial case of the 1-body problem), the differential equations are exactly solvable. When the problem is generalized to three or more bodies, it becomes impossible (with the exception of highly specialized cases) to find closed-form solutions to the differential equations which define the motion of the system. It is therefore impossible to exactly determine where a given body will be after any finite period of time. It is, however, possible to use the differential equations which define the system to approximate positions of the particles through iteration over a small time-step.

An example of the n-body problem is our Solar System. The planets, and the Sun all exert a gravitational attraction on each other, which creates the dynamics which we observe. Now, the particle which we were observing in the above examples was not attracting any other objects to it, but instead, was moving about according to the forces defined by certain fixed objects. An example of this would be a satellite sent into the Solar System, with no power of its own. Its mass is so small that it would not noticeably affect that position of the planets or the Sun, but the planets and Sun would most definitely have an impact upon the satellite's trajectory. Though the Sun and planets are themselves in motion, and not fixed as the points in our system are, but the mathematical foundations remain.

Another, and perhaps better analogy lies in the motions of a specialized pendulum. Consider a string tied to a ceiling with a magnetic ball attached to the end (a basic pendulum, essentially). Now, consider that a number of magnets (all of equal strength) have been distributed about the floor. Electromagnetic force works on the same principle as gravity, with a different property, charge, replacing mass, and a different constant, though the basic governing equation is the same. If one were to release the pendulum, and let is swing for a while, one would notice that the pendulum follows an erratic path, curving toward the various magnets as it moved. This system illustrates the nature of the single object moving under the influence of fixed bodies. This example too has minor flaws (the addition of the gravitational force on top of the electromagnetic force), but is meant to illustrate the basic principles of the system used in our encryption algorithm.

IV. Description & Procedure

A. Simple Encryption

The simple encryption program that we have created encrypts a text file using pseudo random numbers exclusive-or'ed with the original character then placed into the new encrypted text file. This method is relatively easy to crack given that you have the code used to encrypt the file. From there you must create a method that tries many starting random numbers until it finds the one that was originally used. The bonus to this method is that it is very fast. If your algorithm stays secret the code should be relatively hard to break. However, we must assume that the person attacking the code has all the information including the algorithm used to encrypt, but only missing the original data. Thus this method would be fine for personal use, but not for corporate use.

B. Java Program

The Java program will be used to create a nice user interface, since C++ is difficult to work with for such graphics. The Java program will take the user input and save it into a file, which C++ will read. Items that will be input will be the name of the file to encrypt, where to save it, and the coordinates to set up the N-Body system. This allows for each user to customize the system that they use to encrypt their files. We will also be using Java to create a nice visualization of the actual process of encrypting the text.

C. N-Body Algorithm

The fractal of choice for this project is a chaotic system. An example of a chaotic system is the n-body problem. The n-body problem tries to determine the motion of particles in the presence of strongly attractive forces (such as gravitational force and electromagnetic force). Using the principles mentioned above, our algorithm will use a series of simple unary functions operating directly on binary data to render it theoretically undecipherable.

The complexity of breaking this system lies within the difficulty of finding a set of attracting forces that would yield a given path for a particle. For a single path (as defined within the precision of a computers computing ability) there will not be unique solutions, and therefore it will be difficult for an attacker to determine the correct solution (key and corresponding plaintext-ciphertext pair) to the algorithm.

Our primary encryption algorithm is designed to make use of the fractal properties of a chaotic system (an example of which would be the n-body problem), while still maintaining a reasonably straightforward approach to minimize potential security holes, and reduce errors involved in the implementation of the algorithm.

Many considerations had to be taken into account when designing this algorithm. First, the program must use the data it is encrypting as well as a key to determine what operations must be performed on the block of data. Secondly, all functions used within the algorithm must be uniquely reversible. That is there must exist a one-to-one correspondence between the plaintext space and the ciphertext space. Thirdly, there must exist some method based upon the key of determining how data is to be used within the

N-Body Encryption Algorithm

Encrypts a message based upon a series of permutation functions, keyed into an instance of the n-body problem which is simulated for each block of plaintext interpreted as a point in space.

Definitions

- A chaotic system (n-body system) A
- A set of parameters of size n (i.e. location of all equal mass objects) P called points
- P is also the key
- A set of n reversible functions F[k] and their inverses ~F[k]

Encryption Algorithm

- Input P into A
- Sequentially number all points in P
- Loop while there is still some unencrypted data
- Convert a block of data into a point R
- Place R in the chaotic system
- Determine which point in P the point R will be attracted to called K
- Create a ciphertext block consisting of K
- Append to the ciphertext block F[k](R)
- Continue with the loop

algorithm, and eventually, what will be done with it. The algorithm works as follows (also in Figure 1):

Efficiency is a major concern for our algorithm. Although the individual encryption functions used to directly modify data are relatively simple and efficient, finding solutions to the chaotic system can present a computational problem. To minimize time consumed by this part of the algorithm, care has been taken to make the chaotic system as computationally inexpensive as possible (with the primary goal time optimization, and the secondary goal is memory space optimization). It is not necessary for the chaotic system to mimic any particular natural system, but it must have the same basic properties.

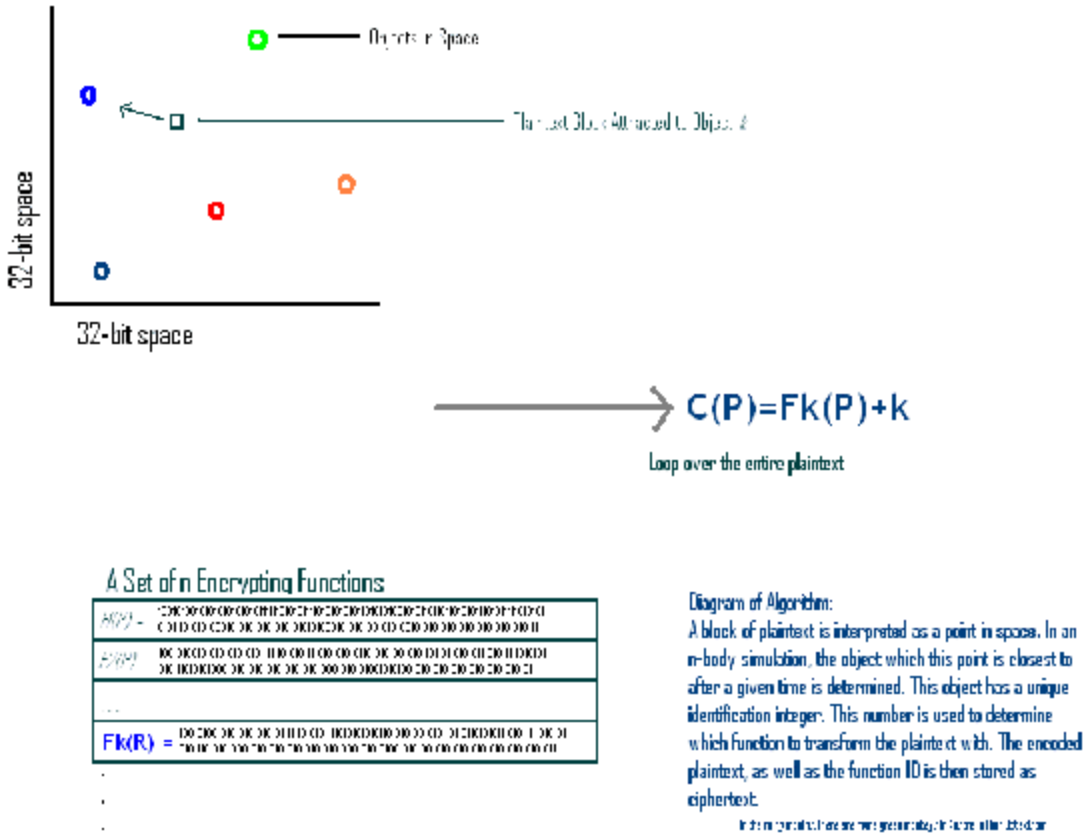
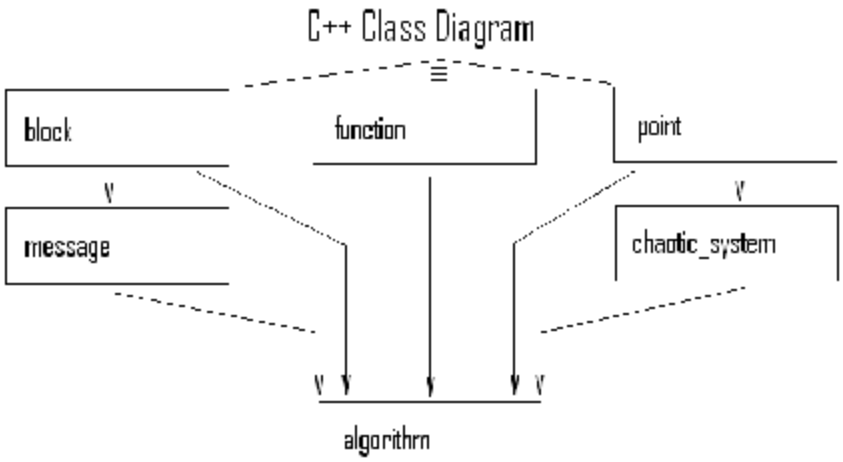


Figure 1

- ***Implementation in C++***

For the implementation of the encryption algorithms we are using, we have decided to implement them in C++ primarily for efficiency, as well as for the familiarity of the language in numerical applications. The class hierarchy for our primary algorithm calls for several small helper classes to assist in the computations for the primary algorithm class (Figure 2).



Note:
The "block" and "point" classes are equivalent to each other.
The "algorithm" class uses all classes.

Figure 2

V. Discussion, Results & Conclusions

A. General Conclusions

We formed our conclusions based upon the data that we gathered. From the data it appears that the original text is sufficiently scrambled so as to flout any person trying to break the code using a brute force method. The simple encryption would work moderately well so long as the attacker does not have the code, but with the code it is only a matter of hours before they could crack the code.

Thus the easiest way to have a somewhat secure code is to have the algorithm kept secret. This would lengthen the life of your code. If you are worried about corporate workplaces where many different people will be using the code and the possibility of the code being leaked is high, you must create a code that would stay secure even if the attacker has the algorithm used to encrypt the information.

B. Advantages & Disadvantages: A Comparison of Methods

Encryption using a chaotic system benefits from the fact that it has been proven that the differential equations involved in the n-body problem have no closed-form solution. Additionally, using parameterized permutation functions means that an attacker will not be able to exploit a patterns in a large sections of ciphertext. Additionally, the attacker will almost be unable to deduce the set of parameters for the n-body problem itself, even if he had access to ciphertext-plaintext pairs.

This method of encryption does have some drawbacks. One is that it is a symmetric algorithm, which means that key exchange must be handled with a specialized protocol or algorithm. Additionally, the use of floating point arithmetic is slower than integer

arithmetic (although the behavior of floating point numbers as continuous-like variables provides much more variation than does integer arithmetic), and is machine dependent. Although it can be handled through the software to ensure uniformity, this would again slow down the process of encryption, as each encrypted block requires thousands of floating point operations. This is still feasible for use on normal workstations, and higher-performance computers, but would make implementation on specialized hardware for encrypting data in real-time infeasible as it would require a significant amount of buffer space as it ran the n-body simulations. The decryption process does not require any floating point arithmetic, however, and therefore specialized real-time decrypting would be very viable.

VI. Acknowledgements

Jim Mims – Computer science teacher, and Science Fair sponsor. Mims has provided assistance in obtaining the required materials, and has arranged for presentations relating to our topic.

VII. References

- Bolotashvili, Givi. Solution of the Linear Ordering Problem (NP=P). < http://arxiv.org/abs/cs.CC/0303008 >
- Menezes, Alred, Paul Van Oorschot, and Scott Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.
- Peitgen, Heinz-Otto, Dietmar Saupe. Chaos & Fractals: New Frontiers of Science. Springer Verlag, 1992.
- Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. John Wiley & Sons, 1995.
- Stinson, Douglas. Cryptography: Theory & Practice, Second Edition. Chapman & Hall, 2002.