Dragon Crypto - A new crypto system

Awnon K. Bhowmik
Department of Mathematics and Computer Science
CUNY York College
awnon.bhowmik@yorkmail.cuny.edu

Unnikrishnan R. Menon
Department of Electrical and Electronics
Vellore Institute of Technology
unnikrishnanr.menon2017@vitstudent.ac.in

Abstract

In recent years cyber-attacks are continuously developing, therefore security specialists must stay busy in the lab inventing new schemes to keep attackers at bay. In this paper we outline the details of a new crypto system based on the dragon curve fractal. The security level of this scheme is based on multiple private keys, that are crucial for effective encryption and decryption of data.

1 Introduction

Fractals are just a self replication of a particular pattern. There are many fractals that are found in nature. One such fractal curve is known as *Heighway Dragon*, or simply the *Dragon Curve*. Because this is a fractal, hence it requires an IFS (Iterated Function System). In simple words, we make a simple pattern, put it into a recursive function, before running a loop and calling the function over and over again until our objective is accomplished.

This dragon curve fractal, or its generation algorithm forms the basis of this cryptosystem, and hence the proposed name *dragon crypto*. Here, we take an arbitrary string of characters, pass each character through the *Koblitz encoder* to obtain the starting point for the IFS.

2 The Dragon Fractal

A fractal is a repetition of an initial geometric shape. This is generated by something known as the Iterative Function System (IFS). The dragon curve is one such fractal. There are multiple ways to generate this fractal.

2.1 Generating the Dragon Fractal

Given an initial length l and n number of iterations, the following steps combined, generates the curve...

-

3 Koblitz Encoding Algorithm

The encoding algorithm is as follows...

- Given a message M, convert each character m_k into a number a_k using Unicode, where $b = 2^{16}$ and $0 < a_k < 2^{16}$

- Convert the message M into an integer using

$$m = \sum_{k=1}^{n} a_k b^{k-1}$$

. In practice we choose an n to be less than or equal to 160 such that m satisfies $m \leq 2^{16\cdot 160} < p$.

- Fix a number d such that $d \leq \frac{p}{m}$. In practice we choose the prime p large enough so that we can allow $d = 100$.

- For integers $j = 0, 1, 2, \cdots d - 1$ we do the following loop
  - Compute the x coordinate of a point on the elliptic curve as $x_j = (dm + j) \mod p$ where $m = \left\lfloor \frac{x_j}{d} \right\rfloor$
  - Compute $s_j = (x_j^3 + Ax + B) \mod p$
  - If $(s_j)^{\frac{p+1}{2}} \equiv s_j \mod p$, then define y coordinate of a point on the elliptic curve as $y_j = (s_j)^{\frac{p+1}{4}} \mod p$. Return the point (x_j, y_j).

Thus we are able to encode our message M, as an element of the abelian group $G = E(\mathbb{F}_p)$ [9]

4 Trapdoor Function

A trapdoor function is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction without special information, called the *trapdoor*. Trapdoor functions came to prominence in cryptography in the mid-1970s with the publication of asymmetric (or public-key) encryption techniques. Several function classes have been proposed, and it soon became obvious that trapdoor functions are harder to find than was initially thought.

The main novelty of our cryptosystem is the use of **The Heighway Dragon Fractal** as a trapdoor function. The algorithm starts off with a secret message that needs to be encrypted (called the plainText). For instance, let's consider the message "Hello World".

The Koblitz encoder accepts the plainText along with the curve parameters (obtained from the private key). The koblitz encoder spits out an encoded point in 2D Cartesian Space for each character present in the plainText with the help of curve parameters.These points are now the starting point of the Dragon Curve Fractals. For each character, a corresponding startng point and a dragon fractal are generated.

Now the components of the private key (including size,iterations,angle) are used to generate the the fractal for each character from their corresponding starting points.

Private key parameters

- The size defines the length of each forward step.
- The Iterations defines the number of iterations for generating the fractal.
- The Angle defines the starting angle for the fractal.

Once the fractals are generated for each encoded starting points, the corresponding ending points are noted and stored.

$$p_{\text{start}} = \{(x_i, y_i)\} \qquad i = 1, 2, \cdots n$$
$$p_{\text{end}} = \{(x_j, y_j)\} \qquad j = 1, 2, \cdots n$$