

CHAOS AND ENCRYPTION

by Dr. Stephen B. Soffer

Copyright 2007 Stephen B. Soffer and
SBS Publications & Educational Services

This material may be used within your classroom, school or home schooling parent's home. It is not to be copied or distributed outside of these.

Note: Sample pagination differs somewhat from that in the full document.

Level: From ninth grade up – for any students who are familiar with use of a graphing calculator.

Purpose: To introduce students to the use of chaos to hide messages from spies and an introduction to cryptography.

Companion Lesson Plan: *Chaos – An Introduction – Learning by Doing*

INTRODUCTION

In a companion lesson plan, *Chaos – An Introduction – Learning by Doing*, I introduced instructors wishing more background to the field of **chaos** or *nonlinear dynamics*. Here the focus is on one of its applications, **encryption**, sometimes called *coding*, although we will see that these are really somewhat different concepts. These terms imply the companion processes of **decryption** and *decoding*. The goal is to suggest some ways of using the *apparently* random nature of nonlinear sequences to encrypt messages in order to prevent unauthorized people from getting their true content, even if they intercept the entire encrypted message.

As part of this process, I will suggest and give examples of *tests* to see how effective is the encryption method from preventing unauthorized decryption. A graphing calculator with *sequential* as well as *function* modes (such as the TI-83 Plus® by *Texas Instruments*) is sufficient to get the basic ideas and illustrate them. (A powerful computer would be needed for more serious work in the field.)

At this point, I will confess to being a non-expert in the field of cryptography. An expert in the field may find the approach here naive and possibly subject to breaches in a real world system.

I will start with a brief review of **nonlinear dynamics** – the technically more accurate name than *chaos*. (The companion lesson plan gives a more detailed discussion.) Then I will propose a way to use the apparently random, or “noisy”, properties of nonlinear dynamical sequences for encrypting and decrypting messages. These will be tested for feasibility in two ways:

1. The desired recipient of the encrypted message can easily decrypt it and recover the original message.
2. An unintended recipient who intercepts the encrypted message – a “spy” – cannot easily recover the original message.

The second requirement will include tests of *sensitivity*. By that I mean how close the spy can get to the encryption “*keys*” and still not get the message.

Towards the end, I will go into some additional background on encryption and decryption, including some on **public-key cryptography**, which is a popular method used in practice. This will include references to text and online resources.

ENCRYPTION VS. CODING

I mentioned that *encryption* and *coding* are somewhat different terms. Most messages consist of characters and symbols in a standard language, such as – but not limited – to English. In English, this might include the 26 letters of the alphabet (52 if upper and lower case letters are considered distinct), 10 digits: 0,1,2,...,9, the space, and various symbols, such as the comma (,), period or decimal point (.), exclamation point (!), etc. **Coding** is the conversion of these characters and symbols of what is called **plaintext** to other forms, usually numeric. This could mean that each character or symbol is assigned a two digit nonzero integer. Normally, no attempt is made to make coding hard to break. It is used to set up for the next step – *encryption*. Encryption is the process of making the coded message hard to decipher unless the recipient knows one or more values called *keys*. As a very simple, naive system, coding the 26 letters might be assigning the values 11 to 36 to them (leaving 0 to 9 for the digits), and adding 5 to each. For example, A with a coded value 11 would become $11 + 5 = 16$, or the code for F. W, with a coded value of 33 would become 12 or B’s value, since 38 is past the end of the alphabet codes. In other words, we have a modulus 26 system. Decryption and decoding are the reverse processes to recover the original plaintext.

BASICS OF NONLINEAR DYNAMICS

We can start with a sequence that represents a simple model for the growth of a population (and also the growth of a bank account using compound interest). Let

P_0 be the population in whatever year researchers start to track it. If the population is to grow, we can take next year's population as

$$P_1 = rP_0$$

where $r > 1$ is a real constant. Obviously, if there is sexual reproduction, two partners are needed and $r > 2$ is needed for growth, even more to allow for deaths before reaching reproductive age. If we let subscripted P denote the number of reproductive pairs, we can use $r > 1$ as needed for growth. If this pattern continues at the same reproductive rate over succeeding years

$$P_2 = rP_1$$

$$P_3 = rP_2$$

etc. In general,

$$P_{n+1} = rP_n, n = 0, 1, 2, 3, \dots$$

It is easily seen that

$$P_n = r^n P_0, n = 0, 1, 2, 3, \dots$$

This is a model of *exponential growth* as long as $r > 1$.

Its behavior is very regular and predictable. Also, from the standpoint of encryption, anyone observing the behavior over a few iterations, $P_0, P_1, P_2, P_3, \dots$ would quickly infer the pattern is given by $P_{n+1} = rP_n$. From the standpoint of chaos, it is important to note that this is a **linear** relationship between succeeding populations. This should not be confused with the fact that plotted against the index n , it is exponential.

Now let's get more realistic. Writers have pointed to the finite nature of resources to support a population, especially if in a closed or isolated ecosystem (say a remote valley). The exponential growth model predicts the population can expand without limit. This must break down eventually, although it may hold for a few earlier generations, when resources are plentiful.

One popular way to impose limits caused by finite resources is the **logistic parabola**

$$P_{n+1} = rP_n \left(1 - \frac{P_n}{P_{\max}} \right), n = 0, 1, 2, 3, \dots$$

where P_{\max} is the largest population of a species that can be supported. If it

reaches that value, it will become extinct.

A more compact form comes from defining

$$x_n = P_n / P_{\max}$$

The model is then recast as

$$x_{n+1} = rx_n(1 - x_n)$$

It follows that $x_n = 0$ or $x_n = 1$ signals the end of the line. Then there is no further reproduction, so $x_{n+1} = 0$.

A notable fact about the new model is that the model relating succeeding generations is **nonlinear**. It can be expressed as

$$x_{n+1} = rx_n - rx_n^2$$

This shows its *parabolic* form. It also shows that, for a population that is very small compared to its maximum possible value, such that $x_n \ll 1$, it follows that $x_n^2 \ll x_n$. In this case, the second term is negligible in size, and we recover the exponential growth model. This reflects the lack of pressure on the small population that only requires a small fraction of the available resources.

For such a model to be realistic, we require

$$0 \leq x_i \leq 1$$

It is easy to show (see the companion lesson plan) that this puts limits on the model parameter, requiring

$$0 < r \leq 4$$

The behavior of this relatively simple model (and other simple nonlinear ones) is surprisingly complex. Here, I will give a brief indication of this, looking at intervals of r .

$r < 1$: First consider a value of the model parameter that is less than one and see what happens for three values of starting relative population as in the table. Values are rounded to five significant figures.

r	x_0	n	x_n
0.9	0.2	0	0.2
		1	0.144
		2	0.11094
		3	0.088767
	
		10	0.028447
	
50	3.2740×10^{-4}		
0.9	0.5	0	0.5
		1	0.225
		2	0.15694
		3	0.11908
	
		10	0.034046
	
50	3.7523×10^{-4}		
0.9	0.8	0	0.8
		1	0.144
		2	0.11094
		3	0.088767
	
		10	0.028447
	
50	3.2740×10^{-4}		

From this sample of sequences, we learn a couple of things.

Note that starting values of 0.2 and 0.8 give the same sequence. This is because the model is symmetric about 0.5, as can be seen by plotting the parabola or by analysis.

More importantly, we note that, regardless of the starting value, the sequence seems to tend towards zero. Also, later values ($n = 10, 50$) are similar for all of these starting points. I will jump to a general statement.

r	x_0	n	x_n
3.831	0.5	8	0.15507
		9	0.50195
		10	0.95774
		11	0.15507
		12	0.50196
		13	0.95774

The case shown above is picked from the left side of the window shown in the last figure. It shows that the orbit is quickly trapped in a *period-3* orbit, which continues for larger n :

$$0.15507 \rightarrow 0.50196 \rightarrow 0.95774 \rightarrow 0.15507, \dots$$

The next case is from deep inside the chaotic region.

r	x_0	n	x_n
3.95	0.5	0	0.5
		1	0.9875
		2	0.04876
		3	0.18320
		4	0.59108
		5	0.95474
		6	0.17070
	
		50	0.44557
		51	0.97580
		52	0.09329
		53	0.33412
		54	0.87881
		55	0.42067
		56	0.96264

We can see no clear pattern initially or after 50 iterations. There is no sign of anything other than *randomness*.

PROPOSED ENCRYPTION SCHEME USING THE LOGISTIC PARABOLA MODEL

The main point behind the proposal is that – although a *chaotic* sequence is *completely determined* by specifying the model, parameter(s), and starting value (strictly at any given computer or calculator accuracy) – **reversing** the process is **very difficult**. By this I mean: based on a detected chaotic sequence $x_0, x_1, x_2, x_3, \dots$, it is very hard to determine the model parameter, even if you know that the model used to generate the sequence is, say, the logistic parabola. From now on, I will stay with the logistic parabola.

This leads to making the parameter r and starting value x_0 part of a **key** for *encryption* and *decryption*. Note that this requires that r be in the chaotic region, generally, $r > 3.7$, *but* excluding windows of the type we saw above, where the behavior is simpler.

Using x_n to Add “Noise” to the Signal

actually be *counterproductive* here. It gives the orbit more chance to settle down to the attracting cycle. Some of the earlier iterations are shown next.

n	x_n
0	0.7235
1	0.76618
2	0.68613
3	0.82481
4	0.55343
5	0.94657
6	0.19372

So far, there is no suggestion of a 3-cycle. Further along, we get the following.

n	x_n
12	0.84211
13	0.50923
14	0.95717
15	0.15700
16	0.50690
17	0.95732
18	0.15650
19	0.50558
20	0.95738

After some initial random wandering, the cycle is now starting to become established.

SOME ADDITIONAL BACKGROUND ON ENCRYPTION/DECRYPTION

Encryption is closely related to the process of **decryption**. The intended recipient of the encoded and encrypted message must be able to recover its original content. At the same time, it is necessary to make it impossible, or at least very difficult, for an unintended recipient to **decrypt** it, (or possibly even to distort the material). This

is especially important in the era of global communication via satellites and using international communications (Internet, mobile telephones, etc.).

Here, I will draw from some sources and give some additional background on *encryption/decryption* in general, **public-key cryptography**, and some on the use of chaos (some of which is generated by physical devices as well as mathematically). Two of my main sources are listed in the References. One is a book, **Public-Key Cryptography**. The other is from articles in the **CRC Concise Encyclopedia of Mathematics**. Parenthesized references here are to PKC and CEM, with page numbers and for subtitle information. The first of these references is useful for background on cryptography in general.

First, I have used the term **cryptography**. This (PKC, p. 2) is a general word for all activities involving converting messages to *encrypted* forms for transmission (once by courier, now mostly electronically). It also includes *decryption* by the “legal” recipient and attempts of the unintended recipient to “break the code”.

First let’s get one bit of terminology straightened out: *encoding* vs. *encryption* and *decoding* vs. *decryption*. In order to do this, we need a more basic definition: **plaintext**. This (PKC, p. 2) is the *original message*. It is written in a *natural language* – meaning one of the world languages – usually in *alphabetic* form, with numbers and various symbols (including the blank or empty symbol of the space). [In computer lingo, this is often called *alphanumeric* representation.] Many of the world’s languages use the same notation as English, with lower case letters a, b, c, ... and upper case letters A, B, C., Generally upper case or lower case letters alone suffice for messages. Other languages (Russian, Hebrew and Greek for example) use roughly analogous symbols for their alphabets. In Greek for example, alpha = α is equivalent to our a, beta = β is equivalent to our b, etc. However, languages using pictographs, such as Chinese, are more difficult to translate into symbols.

Encoding denotes the conversion of the plaintext into some other form for transmission (PKC,p. 4). [Note: In some of the material following, PKC tends to be rather formal. In my presentation, I try to keep the language simpler.] The result need not be alphabetic. It can also be *numeric*. And, if numeric, it does not have to be in *decimal* format. Since computerized information is basically *binary* (in *bits*, which is short for *binary digits*), one can encode the 26 letters of the English

alphabet by five bit numbers. Some of these are shown.

$$A \rightarrow 00001, B \rightarrow 00010, C \rightarrow 00011, \dots, Z \rightarrow 11010$$

Checking the case of Z, converting to decimal gives

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 16 + 8 + 0 + 2 + 0 = 26$$

This agrees with Z being the twenty sixth letter of the alphabet.

In the encoding stage, there is usually no serious attempt to hide the meaning of the original plaintext message.

Encryption is the conversion of the encoded plaintext to a form that is (hopefully) only easy to decipher (decrypt) by the intended or “legal” receiver. It usually involves one or more **keys**. I gave examples in the chaotic encryption scheme above. You will see more about keys shortly.

The main goal of encryption should be that it is *easy to encrypt*, but *hard to decrypt* for a recipient who does not have the complete set of keys (PKC, p. 4-5).

An elementary of encoding – and easily broken – is simple substitution by switching each letter of the plaintext to another letter that is later or sooner, often by the same number of characters. The

