Logistic Encryptor

Nicholas Funari Voltani *

December 4, 2020

I wrote this program totally inspired on Baptista's article "Cryptography with Chaos"[1], in which the author suggests a cryptographic method using the logistic equation. Its most interesting feature is that it takes full advantage of this equation's chaotic behavior (sensitivity on initial conditions) for security implementations.

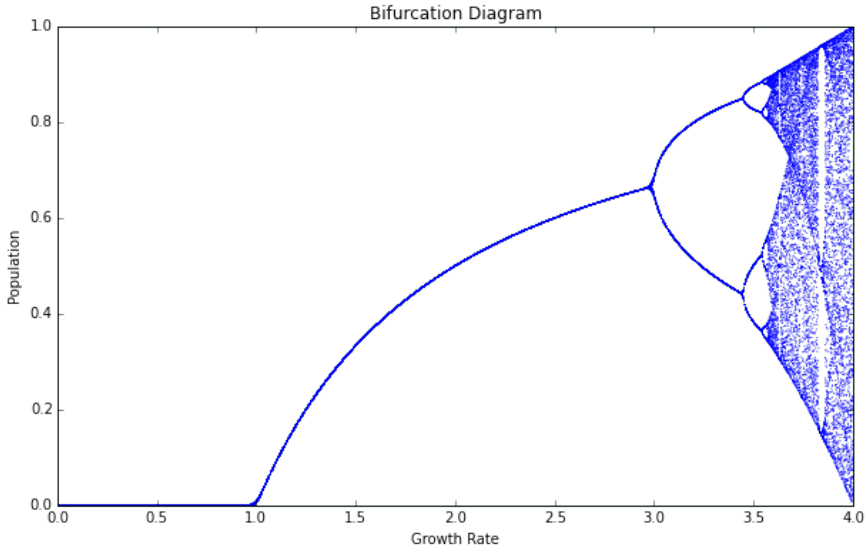
However, it isn't a completely independent cryptographic method, as it relies on sending, alongside the encrypted message, the initial conditions (keys) used for the encryption, which would require some other (independent) form of cryptography. Still, an interesting read, and proposal, regardless.

1 The logistic equation

We start with a discrete-time logistic equation, as below

$$\begin{aligned} x_{n+1} &= b\, x_n\, (1 - x_n) \\ &\equiv F(x_n) \end{aligned}$$

The algorithm will take advantage of its chaotic attractors, which can be seen on its bifurcation diagram, as below:



Blue dots represent fixed points of the equation, which appear after many consecutive iterations of F applied on some initial condition X_0. When we have a finite number N of points vertically above some x-axis value, we say we have a stable periodic orbit (of period N). For $1 \leq b \leq 3$, we have period 1; for $3 \leq b \leq 3.5$, we have period 2, then period 4, etc, until chaos ensues at around 3.6 (we call it a period-doubling cascade/bifurcation).

*nicholas.voltani@usp.br

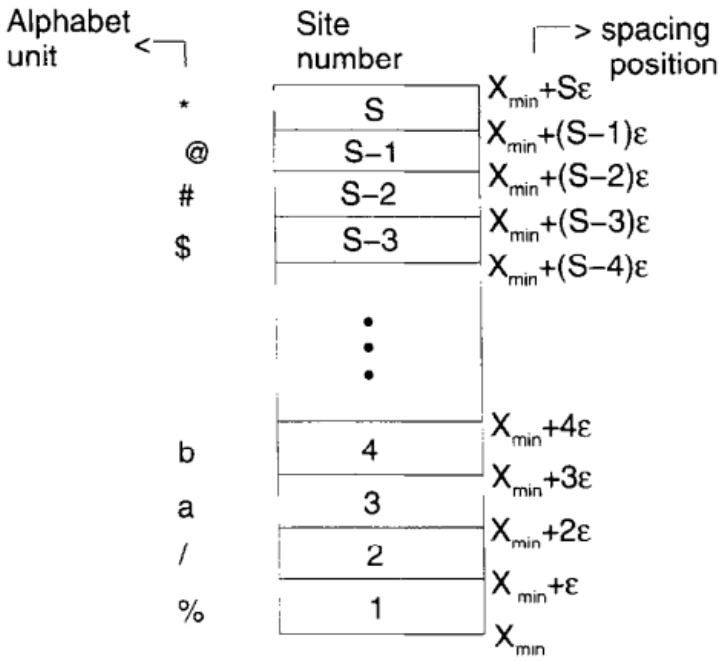
In particular, we will take some specific values of b (x-axis) such that we have a continuous vertical line; that is to say, it corresponds to b's chaotic attractor. I assume it's preferable to use values greater than 3.7 (and less than 4), since its chaotic attractor won't be disconnected, but maybe it doesn't really make a difference; it just needs to be a chaotic attractor.

2 The Encryption

To write the message, we need some prior choice of letters (our alphabet S, of size $|S|$). Given a chaotic attractor (X_{min}, X_{max}), we partition it in $|S|$ equal subintervals of size

$$\epsilon = \frac{X_{max} - X_{min}}{|S|}$$

each of which will correspond to its respective letter of the alphabet. A representation from Baptista's article is shown below:



So to encrypt some message M, we need to have the following information (which are also called the "keys" of the encryption process):

- An alphabet S in which M was written;
- a logistic (chaotic) parameter b, i.e. its chaotic attractor (X_{min}, X_{max});
- some initial condition X_0 from which to iterate the logistic function F.

To take full advantage of b's chaotic behavior, we can also assume that

- some transient time N_0

is also a key; that is, we iterate the logistic equation, starting from X_0, for N_0 timesteps, and *then* we start the encryption *per se*. A pseudocode of the method is presented in the appendix.

A variant of the algorithm above, also in the appendix, is to introduce a parameter η such that, each time we find some N_k to encrypt the character p_k, we generate a random number κ, and only accept N_k if

$\eta \geq \kappa$; else, we keep iterating until we reach V_{p_k} again, repeating this process until $\eta \geq \kappa$. This allows for a quicker way to change the cryptographic keys, since only the sender will know η, not the receiver.

To decrypt an encrypted message e, we need the keys (S, X_0, b, N_0) to iterate X_0 for each integer N_k (after the transient N_0), to see in which interval X_0^k falls into[1], to infer the original plaintext character p_k.

References

[1] BAPTISTA, M. S. Cryptography with chaos. Physics letters A, v. 240, n. 1-2, p. 50-54, 1998.

A Algorithms

Algorithm 1 Encrypt message $M = (c_1, c_2, \ldots, c_m)$, $\{c_k\}_{k=1}^m \subset S$, with keys $(S,\ b,\ X_0, N_0)$

Initial conditions
$X = X_0$
(l, r) = ChaoticAttractor(b)
Partition (l, r) into $|S|$ subintervals (denoting the letter c's respective interval as V_c)

Remove transient
for $i = 1, \ldots, N_0,$ **do**
  $X = F(X)$ # Iterate logistic equation
end for

$e = ()$ # Create to-be-encrypted message e, as an empty list
for $c_k \in M$ $(k = 1, \ldots, m)$, **do**
  repeat
    X = F(X) # Iterate logistic equation
  until $X \in V_{c_k}$
  Write total time N_k taken to reach V_{c_k}
  Concatenate it to list e
end for

return e

[1] where $X_0^k \equiv F^{N_k}(\ldots(F^{N_1}(F^{N_0}(X_0)))\ldots)$.

Algorithm 2 Encrypt message $M = (c_1, c_2, \ldots, c_m)$, $\{c_k\}_{k=1}^{m} \subset S$, with keys ($S$, b, X_0, N_0, η)

Initial conditions
$X = X_0$
(l, r) = ChaoticAttractor(b)
Partition (l, r) into $|S|$ subintervals (denoting the letter c's respective interval as V_c)

Remove transient
for $i = 1, \ldots, N_0,$ **do**
  $X = F(X)$ # Iterate logistic equation
end for

$e = ()$ # Create to-be-encrypted message e, as an empty list
for $c_k \in M$ $(k = 1, \ldots, m)$, **do**
  repeat
    X = F(X) # Iterate logistic equation
  until $X \in V_{c_k}$ and $\eta > \kappa$ ($\kappa \sim \mathcal{N}(1, 0)$)
  Write total time N_k taken to reach V_{c_k}
  Concatenate it to list e
end for

return e

Algorithm 3 Decrypt message $e = (N_1, N_2, \ldots, N_m)$, $\{e_k\}_{k=1}^{m} \subset S$, with keys ($S$, b, X_0, N_0)

Initial conditions
$X = X_0$
(l, r) = ChaoticAttractor(b)
Partition (l, r) into $|S|$ subintervals (denoting the letter p's respective interval as V_p)

Remove transient
for $i = 1, \ldots, N_0,$ **do**
  $X = F(X)$ # Iterate logistic equation
end for

$\tilde{M} = ()$ # Initialize recreated message $\tilde{M}$ as an empty list
for $k = 1, \ldots, m,$ **do**
  ## Reminder that N_k is total time needed, from very first iteration, up to V_{p_k}'s interval
  for $n = 1, \ldots, (N_k - N_{k-1}),$ **do**
    X = F(X) # Iterate logistic equation
  end for
  Check in which interval $X \in V_c$
  Concatenate its character to list $\tilde{M}$
end for

return $\tilde{M}$