**One-Time Pad Scheme Using Chaos Equation to Generate Key**

**Abstract:** Information security is an important topic in this digital world. Effective cryptographic methods are needed to protect sensitive information communicated through a network. One such cryptographic technique is the One Time Pad (OTP). Introduced in the late nineteenth century and widely used in the twentieth century, this unbreakable symmetric technique requires a key of length minimum as of the message to be encrypted. Sharing a randomly generated Pad (key) is the hardest part of this technique. In our work, instead of sharing the key, we generate it at both ends. Using a chaos equation, a random key is generated at both ends. Few parameters of the chaos equation are shared among the users using public-key cryptography like Diffie-Hellman key exchange method. The values generated by the chaos equation are random and not predictable. Using these generated values, a non-repeating random key is formed. This key is used as a pad to encrypt the message using the one-time pad method and the same key is used at the other end to decrypt and obtain the plain message. To secure the information, we use an asymmetric cryptographic algorithm to share the chaos equation which generates a key and it acts as the pad in one-time pad method.

***Keywords****: one-time pad, symmetric, chaos, public-key cryptography.*

**Introduction**

Cryptography has a very significant role in the exchange of sensitive data. So far to safeguard information, many methods were developed such as Substitution ciphers, Feistel cipher, Data Encryption Standard (DES), Advanced Encryption Standard (AES) to name a few. DES was first published in the year 1975 and it was adopted by the U.S government and many industries in the financial sector as an encryption standard. But in the year 1997, DES was proved to be insecure by performing a brute force attack. In the same year, the National Institute of Standards and Technology announced an initiative to choose a successor to DES. In 2001, AES was selected as a replacement for DES. Till date, AES has never been cracked but there is no mathematical proof stating that AES is unbreakable.

One time pad is the only method which has a mathematical proof that it is unbreakable. It is used widely during the world wars by many spies for diplomatic communications. One time pad requires a long key which makes it difficult to use. The key used for one-time pad should be truly random and cannot be reused.

**Related Works**

To overcome the difficulty of using One Time Pad, in the year 1987, a new algorithm called Rivest Cipher 4 (RC4) was published. RC4 generates a repeating pseudo-random key using a relatively short secret key to encrypt the message using OTP. The encrypted output of RC4 is not truly random and this property made it broken.

**Proposed Work**

One Time Pad is the only cryptographic method which is mathematically proved to be unbreakable, but sharing the longer key is the difficult part to use this algorithm. So, to generate a non-repeating key we use chaos theory. Chaos theory is the study of states of apparently-random disorders and irregularities which are governed by initial conditions and simple laws, one of the popular examples is the butterfly effect. Lorenz attractor, double pendulum, topological mixing, period-doubling bifurcation are few of the chaos functions. Chaos equations describe the time dependence of a point in a geometrical space. These points in a geometrical space are used to generate a key. This key is used to encrypt and decrypt the data using the OTP scheme. In our work, we use double-pendulum as a chaos equation. Parameters of this equation are sent by the sender using the Diffie-Hellman key exchange method or both the sender and receiver have the prior information about those parameters.

Since the One-Time pad is unbreakable with a non-repeating long random key, producing such non-repeating long random key using chaos equations makes this method more secure than the existing models. Though there are few algorithms which produce long keys for One Time pad they repeat the short random key to produce the long key, but it is proved that with proper cryptanalysis such methods can be broken. So, the proposed work stands unique among such methods.

**Algorithm**

Alice Bob

Public keys: P, G

Private key: a Private key: b

x = Ga mod P y = Gb mod P

Exchange of generated keys

Secret key generation

ka = ya mod P kb = xb mod P

Both Alice and Bob have symmetric secret key ka = kb

Alice encrypts the parameters of the chaos function using the symmetric secret key and sends the encrypted data to Bob.

Since we use a double pendulum for a chaos function, the parameters shared between Alice and Bob are lengths of rod one and rod two, masses of both the bobs, angles at which both the rods are suspended and the length of the key to be generated. It means that both the users need seven numbers (parameters) to generate the long key.

Now, both Alice and Bob have the parameters of the chaos function.

Using these parameters both Alice and Bob generate the key and use it as a pad for encryption and decryption of the data.

OTP algorithm

for i in range(len(message)):

encrypted\_text = message[i] *xor* Key[i]

for i in range(len(encrypted\_text)):

message = encrypted\_text[i] *xor* key[i]

**Python code for generating Key**

import numpy as np

from scipy.integrate import odeint

import matplotlib.pyplot as plt

from matplotlib.patches import Circle

# Pendulum rod lengths (L1, L2 in meters), bob masses (in kg), rod suspension angles(degrees), Key Length(KeyLength).

L1, L2 = 1.2, 1

m1, m2 = 8, 3

ang1, ang2 = 120, 220

KeyLength = 256

# The above values are considered as parameters shared between Alice and Bob.

def derivative(y, t, L1, L2, m1, m2):

alpha, z1, beta, z2 = y

c, s = np.cos(alpha-beta), np.sin(alpha-beta)

dalpha = z1

dz1 = (m2\*9.81\*np.sin(beta)\*c - m2\*s\*(L1\*z1\*\*2\*c + L2\*z2\*\*2) - (m1+m2)\*9.81\*np.sin(alpha)) / L1 / (m1 + m2\*s\*\*2)

dbeta = z2

dz2 = ((m1+m2)\*(L1\*z1\*\*2\*s - 9.81\*np.sin(beta) + 9.81\*np.sin(alpha)\*c) + m2\*L2\*z2\*\*2\*s\*c) / L2 / (m1 + m2\*s\*\*2)

return dalpha, dz1, dbeta, dz2

# Maximum time, time point spacings and the time grid (all in s).

tmax, dt = KeyLength\*0.1, 0.01

t = np.arange(0, tmax+dt, dt)

# Initial conditions: alpha, dalpha/dt, beta, dbeta/dt.

y0 = np.array([ang1\*np.pi/180, 0, ang2\*np.pi/180, 0])

# Do the numerical integration of the equations of motion

y = odeint(derivative, y0, t, args=(L1, L2, m1, m2))

# Unpack z and theta as a function of time

alpha, beta = y[:,0], y[:,2]

# Convert to Cartesian coordinates of the two bob positions.

x1, y1 = L1 \* np.sin(alpha), -L1 \* np.cos(alpha)

x2, y2 = x1 + L2 \* np.sin(beta), y1 - L2 \* np.cos(beta)

# Plot a trail of the m2 bob's position for the last trail\_secs seconds.

trail\_secs = 10

# This corresponds to max\_trail time points.

max\_trail = int(trail\_secs / dt)

def make\_plot(i):

ax.plot([0, x1[i], x2[i]], [0, y1[i], y2[i]], lw=2, c='k')

c0 = Circle((0, 0), 0.05/2, fc='k', zorder=10)

c1 = Circle((x1[i], y1[i]), 0.05, fc='b', ec='b', zorder=10)

c2 = Circle((x2[i], y2[i]), 0.05, fc='r', ec='r', zorder=10)

ax.add\_patch(c0)

ax.add\_patch(c1)

ax.add\_patch(c2)

ns = 60

s = max\_trail // ns

for j in range(ns):

imin = i - (ns-j)\*s

if imin < 0:

continue

imax = imin + s + 1

ax.plot(x2[imin:imax], y2[imin:imax])

ax.set\_xlim(-L1-L2-0.05, L1+L2+0.05)

ax.set\_ylim(-L1-L2-0.05, L1+L2+0.05)

ax.set\_aspect('equal', adjustable='box')

plt.savefig('frames'+str(i)+'.png'.format(i//di), dpi=96)

plt.cla()

fig = plt.figure(figsize=(5, 5), dpi=96)

ax = fig.add\_subplot(111)

fps = 10

di = int(1/fps/dt)

alphabets = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]

Key = ''

for i in range(0, t.size, di):

print('coordinates: ', x2[i], y2[i])

# X and Y co-ordinates of the bob2 are added and digits from index 8 to 11 are choosen.

key = alphabets[int(str(x2[i] + y2[i])[8:11])%26]

Key += key

print(i // di, 'character of the Key: ', key)

make\_plot(i)

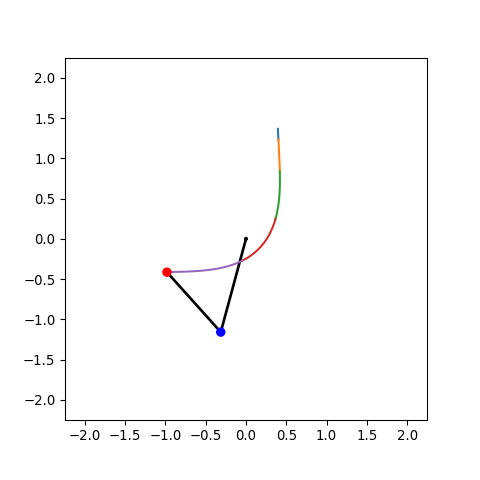
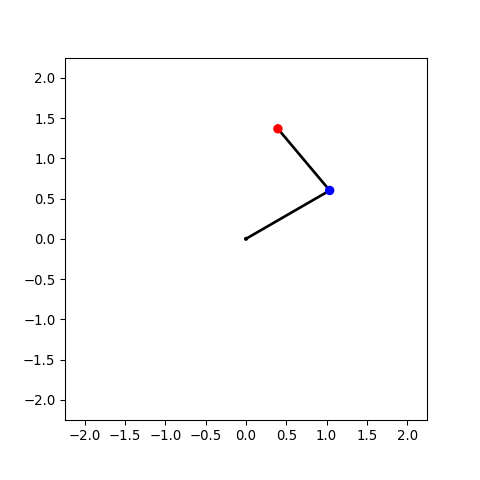
# Final generated key.

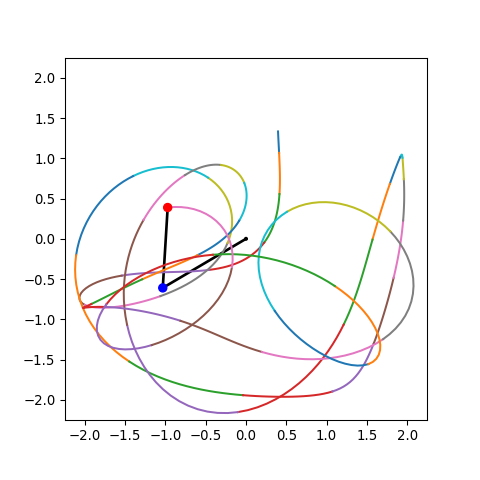
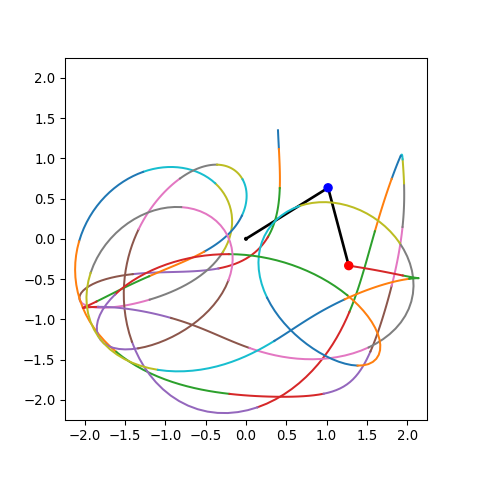
print('Generated Key: ', Key)

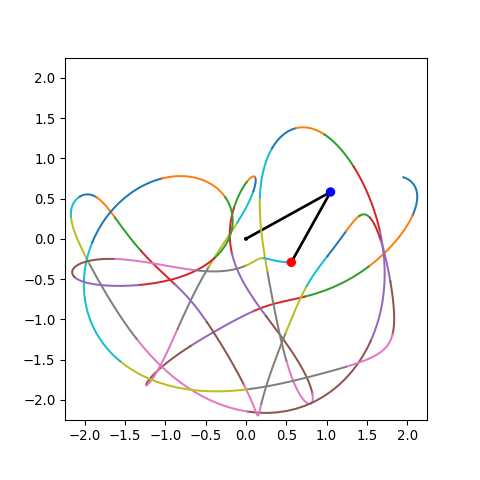
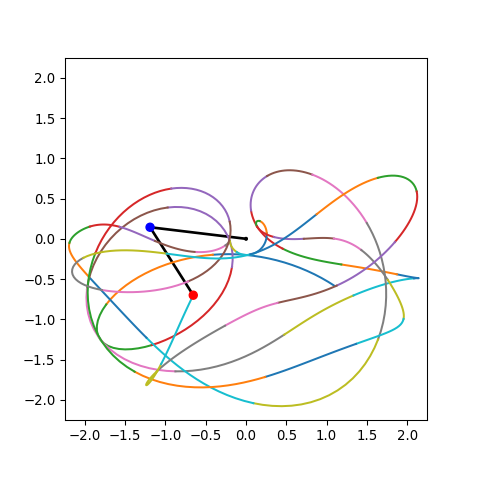
Sample Output

Generated Key:

fzkibvkdbvrysnlsappfydqulqwhnazbkdudthhstzlividbiuiswudatbiiemnyuluoblngsrslmogvynwwckplogqzconpcealpqypblqpkghhlrplwoofjfcpbsilgtovtjkspkngeggyfbcctyrtuuvdqngolrzqabifdkwgqcynuvprbdrmpsyqheofukibodenjcejapeqqgfvixxagwyzulokolosqqjuoujsetrwyzmrzdxbajclaiike

Few output images: (plotted coordinates of the double pendulum)





The above-shown images are a few of the output images. Each character in the key is produced from such images. Key is generated based on the coordinates of the second bob (red). Thus, the obtained key is used as a pad in OTP.

Python Code for key generation and the standalone GUI application generated to encrypt and decrypt the data using one time pad and above key generation algorithm can be downloaded from the below link (Google Drive):

<https://drive.google.com/drive/folders/18HKFn09X6QGEwyd6prT6_TlmyJbh-DNB?usp=sharing>

**Conclusion**

Chaos equations are non-linear and exhibit very complicated phenomena they are highly sensitive to initial conditions which make the long term prediction impossible in general. By making use of this theory along with unbreakable algorithm One Time Pad we designed a new method to encrypt and decrypt the data.

**References**

Menezes. A, Oorschot. P V, Vanstone. S, 1996, Handbook of Applied Cryptography

Borowski.M, Leśniewicz.M, 2012, Modern usage of “old” one-time pad.

Peter I. Kattan, 2012, Chaos Theory Simply Explained.

Carl W. Akerlof, 2012, The Chaotic Motion of a Double Pendulum.

Christian, 2017, The double pendulum. [https://scipython.com/blog/the-double-pendulum](https://scipython.com/blog/the-double-pendulum/)