The Efficacy of Chaotic Neural Networks for Asymmetric Encryption of Audio Files

1st Patrick Pfenning
School of Computing and Data Science
Wentworth Institute of Technology
Boston, MA
pfenningp@wit.edu

***Abstract*—This paper focuses on developing a new algorithm for encrypting audio files using a Chaotic Neural Network (CNN) Said algorithm is benchmarked for its efficiency and effectiveness against two standard encryption methods, Advanced Encryption Standard (AES) and Rivest–Shamir–Adleman (RSA). The intent of each layer of the network utilizes chaotic maps, which can be used to generate pseudo-random arrays. These arrays are transformed into bitmaps which are used to encrypt files using Waveform Audio File Format. The results show that the proposed algorithm is more efficient and effective than both AES and RSA, but may be easier to crack.**

***Index Terms*—chaos, encryption, audio, neural networks, pseudo-random**

I. Introduction

The world we live in has become wildly dependent on the fast transfer of information. Everything, from our work and education to our entertainment and social life, often requires the internet as a medium. With this abundance of information, how can one ensure that their personal data remains private to bad actors? Enter the field of encryption, the process of encoding information. Encryption allows one to take a plaintext message we wish to transmit and transform it into a ciphertext. This ciphertext is illegible to those who do not have a *key*.

Most modern encryption processes create a *psuedo-random* key via a defined algorithm. These algorithms can be broken into two schemes: **Symmetric-key Encryption** and **Public-key Encryption.** Symmetric-key Encryption algorithms create a single secret key which both the sender and receiver have. This key is used to both encrypt and decrypt the message. The security of this method directly depends on the holders of the key as anyone who has it can read the cipher. Public-key Encryption algorithms require the message receiver to generate both a public and private key. The receiver shares the public key with the trusted sender who uses it to encrypt a message. The message is sent to the receiver where the private key is used to decrypt it. Because the private key is the only way to decipher the message, public-keys can be shared with impunity. Though both methods do nothing to prevent the cipher from being intercepted by a third party, if the algorithm is strong enough, it should be illegible.

Billions of packets of information flow throughout the internet on a daily basis. The larger the packet, the longer the encryption will take. In this paper, we will develop our own algorithm to encrypt several audio files of varying sizes. The aforementioned algorithm will be developed using what is known as a **Chaotic Neural Network** (CNN). This algorithm will be benchmarked for both efficiency and effectiveness against one symmetric-key method, **Advanced Encryption Standard** (AES), as well as one public-key method, the **Rivest–Shamir–Adleman** (RSA).

II. Literature Review

A. *Chaos: An Introduction to Dynamical Systems [1]*

Alligood provides an excellent course in the study of such dynamical systems. She successfully explains chaotic phenomena in nature using Linear Algebra, Differential Equations and Numeric Analysis. The book defines chaos as a field of study while introducing the idea of chaotic maps. These maps are recursive in nature, and are highly sensitive to initial conditions. Each iteration is mapped to a new phase-space for which the rate of separation of points in the sequence is directly related to the system's Lyapunov Exponents. There is one exponent for each degree of freedom the chosen system has. The scalar value of the exponent determines how the basis stretch ($LE > 1$) or shrink ($LE < 1$). Sequences created by such maps, though deterministic, can be unstable. These unstable sequences are known as chaotic orbits. This instability is statistically indistinguishable in nature to randomness, making them ideal candidates for providinging repeatable pseudo-random arrays.

B. *A Review on Applications of Chaotic Maps in Pseudo-Random Number Generators and Encryption [2]*

In 2019 Naik and Singh gave a detailed review of Chaotic Neural Networks applications for encryption. The key aspect, as in many of the papers in this section is the creation of a viable generator. One way chaotic maps are used to generate a sequence of length **N**, which is equal to the byte-length of the plaintext file. These values are then converted to binary and one byte are taken from each value, often the first byte after the decimal point. This sequence is then reshaped to match the dimensions of the file being encrypted. The resulting matrices are then XORed with the original file masking the data. If we preform this action a second time, we will revert to the original data. This is extremely useful for the decryption process.

This paper also goes through another chaotic tool, diffusion, the act of swapping indices. The Arnold Cat Map (ACM) [2] provides an efficient way to diffuse desired file prior to encrypting, but is most often used for image encryption only. After several iterations of this map on a matrix the original data is unrecognizable. It should be noted that orbits of ACM are finite, therefore if the map is iterated to the length of the orbit, all diffusion will be undone.

The largest takeaway of this paper is the review of an audio encryption model using sequences from the Henon and Tent maps. Said sequences are XORed to create a secret key. This secret key is then XORed with the audio file to encrypt. Their correlation coefficient between the plain and ciphered files were as low as 0.0014 with entropy was as high as 7.9995.

C. *Comparison of Cryptography by Chaotic Neural Network and by AES [3]*

This 2019 paper by Skovajsová gives a step by step explanation of how AES encryption works and compares it to a prebuilt CNN. Both algorithms were tested on five random images of each of the following sizes: 512b, 1024b, 2048b, 3KB, 30KB and 3MB. For both ciphering and deciphering, the CNN significantly outperformed the AES model in terms of speed. The ciphertext created by both models were identical in size to the plaintext file, showing no degradation of information. It was noted that the CNN used here was a Hopfield network that 1D chaotic maps for its neural weights. 1D maps are much faster for generation, but only require a single initial value. If this initial condition is found by a bad actor, the algorithm will be broken. To counteract this, we will be building multiple layer networks of varying dimensionality, each requiring unique sets initial conditions.

III. Methodology

A. Generalization

Suppose there exists some file A which we wish to encrypt using a CNN. The first layer of our network is to transform $A \to A'$, such that A' is the matrix containing the header bytes of the audio file A [4]. We now define the integer N as the byte length of A'. Each subsequent layer of our network will consist of N nodes [5], with weights generated by the chosen chaotic map. Each map will have the set of initial conditions corresponding to:

1) The chosen parameters of the map.
2) Input variable for each dimension of the map.
3) Some integer k defining the number of iterations to remove from the generator.

We then plug in our parameters, supply our input variables and iterate the map $k + N$ times. Once removing the first k values of the list, we are left with a matrix $Z_{d\times N}$ such that d is the dimensionality of our map. We then apply the following transformation [4], [6] to our matrix:

$$W_i = \bigoplus \left\lfloor |Z_i| \times 10^{10} \right\rfloor \mod 256 \tag{1}$$

Such that i denotes which hidden layer of our network we are on. W_i is the logical XOR to the byte-wise representation of our transformed map outputs. Each hidden layer of our network becomes a column of our overall weighted matrix W such that

$$W = \left[W_1 | \dots | W_n\right]$$

We then XOR A' with all columns of W to obtain our the encrypted header E', and in turn the encrypted audio file E. The matrix W acts as both our encryption and decryption key because

$$A' = A' \oplus W \oplus W = E' \oplus W$$

Theoretically, this methodology should work for any set of chaotic maps we choose. Through the rest of this paper we will develop an algorithm using this methodology with a set of maps.

B. Choosing Our Maps

We have chosen audio as our medium for encryption. For ease of use, we will be using Waveform Audio File Format. Byte representations of these files are one dimensional and thus fit our methodology. Due to the one dimensional nature, we may use any size dimensional map to fit our needs. We have chosen to develop a network consisting of four hidden layers, each representing their own map.

The first layer we will use is the Hénon Map [7]

$$x_{n+1} = 1 - ax_n^2 + y_n \tag{2}$$
$$y_{n+1} = bx_n \tag{3}$$

This is a 1-D map that will act as a *diffusion* layer in our system. Chaotic behavior occurs here with parameters $a = 0.3$ and $b = 1.4$. Our second layer will be the Ikeda Map [4]

$$x_{n+1} = 1 + u(x_n \cos(t_n) - y_n \sin(t_n)) \tag{4}$$
$$y_{n+1} = u(x_n \sin(t_n) + y_n \cos(t_n)) \tag{5}$$
$$t_{n+1} = \beta - \frac{\gamma}{1 + x_{n+1}^2 + y_{n+1}^2} \tag{6}$$

Where this system develops a chaotic attractor whenever $u \geq 0.6$. Adding complexity, we turn to the 3-D map known as the Lorenz attractor [2].

$$\frac{dx}{dt} = \sigma(y - x) \tag{7}$$
$$\frac{dy}{dt} = x(\rho - z) - y \tag{8}$$
$$\frac{dz}{dt} = xy - \beta z \tag{9}$$

We will use parameter values $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$. Finally, we will use the most common chaotic function, the Logistic Map.

$$x_{n+1} = rx_n(1 - x_n) \tag{10}$$

Such that $r = 4$. It should be noted that $x_0 \in [0, 1]$ for chaotic behavior to occur.

IV. Algorithm

A. Key Generation

B. Diffusion

C. Encryption

D. Decryption

E. De-Diffusion

V. Experimentation and Results

Here we test our algorithm against several existing methods

A. Processing Time and Complexity

B. Histogram Analysis

C. Correlation Analysis

D. Peak Signal to Noise Ratio

E. Encryption Quality

F. Vulnerability to Attacks

G. Key Sensitivity

VI. Future Work

VII. Conclusion

VIII. Acknowledgments

References

[1] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos*. Springer, 11 1996.

[2] R. B. Naik and U. Singh, "A review on applications of chaotic maps in pseudo-random number generators and encryption,"

[3] L. Skovajsová, "Comparison of cryptography by chaotic neural network and by aes," in *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo)*, pp. 000029–000032.

[4] B. Stoyanov and T. Ivanova, "Novel implementation of audio encryption using pseudorandom byte generator," *Applied Sciences*, vol. 11, no. 21, 2021.

[5] M. Ahmad and M. Malik, "Design of chaotic neural network based method for cryptographic substitution box," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 864–868.

[6] S. Lokesh and M. R. Kounte, "Chaotic neural network based pseudo-random sequence generator for cryptographic applications," pp. 1–5, 10 2015.

[7] H. Hamdy and A. El-Zoghabi, "Public key cryptography based on chaotic neural network," 08 2014.