Encryption based on Neural Cryptography

Sayantica Pattanayak and Simone A. Ludwig

Department of Computer Science
North Dakota State University
Fargo, ND, USA
{sayantica.pattanayak,simone.ludwig}@ndsu.edu

Abstract. Neural network cryptography is an interesting area of research in the field of computer science. This paper proposes a new model to encrypt/decrypt a secret code using Neural Networks unlike previous private key cryptography model that are based on theoretic number functions. In the first part of the paper, we propose our model and analyze the privacy and security of the model thereby explaining why an attacker with a similar neural network model is unlikely to pose a threat to the system. This proves that the model is pretty secure. In the second part of the paper, we experiment with the neural network model using two different ciphertexts of different length. Parameters of the network that are tested are different learning rates, optimizers and step values. The experimental results show how to enhance the accuracy of our model even further. Furthermore, our proposed model is more efficient and accurate compared to other models for encryption and decryption.

Keywords: encryption, decryption, neural networks

1 Introduction

Historically the major consumer of cryptography were military and intelligence organizations. Nowadays however, cryptography is everywhere. Cryptography has gone from an art form that dealt with secret communication for the military to science that helps to secure systems for ordinary people all across the globe. This also means that cryptography is now a central topic in computer science. Cryptography is furthermore the most significant part of communication security [1]. It maintains the confidentiality that is the core of information security using mathematical techniques.

The confidentiality is maintained with the construction of a ciphertext (now called encryption scheme). The ciphertext provides secret communication between two parties either by symmetric key encryption or asymetric key encryption.

Private-key or the symmetric-key encryption is a setting where two parties share some secret information in advance. The private key encryption consists of an *encryption algorithm*, *Enc*, and takes as input a key k and a plain text message m and outputs a ciphertext C. It is denoted as

$$C = Enc_k(m) \tag{1}$$

The decryption algorithm, Dec, takes as input the same key k and a ciphertext C and outputs the plain text m. It is denoted as

$$m = Dec_k(C) \tag{2}$$

In contrast, the ***Public-key*** encryption or assymetric key encryption is a setting where one party generates a pair of keys (pk,sk) called the public key and the private key, respectively. The public key is used by the sender to encrypt a message for the receiver; the receiver then uses the private key to decrypt the resulting ciphertext. The commonly known RSA algorithm [2], abbreviated after the names of the inventors, is an example of public key encryption. The algorithm implements a public key cryptosystem whose security rests in part of the difficulty in factoring large numbers. The algorithm also permits secure communications to be established without the use of couriers to carry keys and it also permits one to "sign" digitized documents.

Neural Cryptography which is a branch of cryptography is a biologically inspired programming paradigm which enables the computer to learn from observations. In the conventional approach to programming, we tell the computer what to do, breaking big problems up into many small ones; precisely defined tasks that the computer can easily perform. By contrast, in a neural network we do not tell the computer how to solve our problem. Instead, it learns from observational data, figuring out its own solution to the problem at hand.

Automatically learning from data sounds promising. However, until 2006 we did not know how to train neural networks to surpass more traditional approaches, except for a few specialized problems. What changed in 2006 was the discovery of techniques for learning in so-called deep neural networks. These techniques are now known as deep learning. They have been developed further, and today deep neural networks and deep learning achieve outstanding performance on many important problems.

Another example of how neural cryptography is used is in voice recognition as demonstrated in [3]. The paper proposed a technique to reliably generate a cryptographic key from a user's voice while speaking a password. The authors also showed how their technique is sufficiently robust to enable the user to reliably regenerate the key by uttering her password again.

Advancing these lines of work, we propose a model where a neural network can learn to perform encryption and decryption and thereby protecting the communication between two entities. We conducted different experiments on our model in order to identify the best structure and parameters. We experimented with our model with ciphertexts of different length and with different other parameters like network structure, learning rates, optimizers and step values. Our findings show a way to improve the accuracy of the model.

Our paper continues as follows. Section 2 describes the related work in the field of neural cryptography. In Section 3, our proposed model and its security is described. Section 4 includes experiments conducted as well as the results and findings. Finally, we conclude with the conclusion and future work (Section 5).

2 Related Work

Until now, there has been a large number of studies concerned with the usage of neural networks in cryptography. Neural cryptography applications were researched first in [4]. The paper defined a new identification scheme which can be used in smart cards because of small size data and easy operations. They proved how their identification scheme is secure against the most efficient attack known using a technique called simulated annealing.

A symmetric probabilistic encryption scheme based on the chaotic classified properties of Hopfield neural networks was studied in [5]. The authors showed how a discrete Hopfield Neural Network (HNN) model is favourable for neural cryptography. The HNN is usually referred to as an associative memory network because the stable states of the network are in the form of system attractors, which can be used to store patterns and correct error messages by a Minimum Hamming Distance (MHD).

Another chaotic neural networks and its VLSI architecture for digital signal encryption and decryption was proposed in 2000 [6]. Chaotic neural networks are used for encryption as well as for decryption because of high security, no distortion in signal and also suitability of system integration.

Another paper proposed the basis of the Neural Key Exchange protocol [7]. Neural Key Exchange Protocol is based on the synchronization of the weights of a Tree Parity Machine (TPM) [8] similar to the selection of chaotic oscillators in chaos communication. In other words, the knowledge of the output does not uniquely determine the internal representation, so the observer cannot tell which input vector was updated. However, the Neural Key Exchange is vulnerable to three types of attacks [9]. The attacks are Geometric attack [10], Genetic attack [11], and probabilistic analysis [12]. These attacks can be fixed by addition of a feedback mechanism [13].

Authors in [14] proposed an encryption key based Artificial Neural Network (ANN). The plain text message consists of bits. Then, the bits are transmitted to the recipient. The paper proposed a backpropagation network for their proposed approach.

3 Our Proposed Approach

Our neural network model is based on the Backpropagation network [15]. A Backpropagation network is used for supervised learning. The algorithm assumes a feedforward NN architecture.

Our proposed model (Figure 1) is based on symmetric key encryption. Like in cryptography we have Alice and Bob here also we will assume that Alice and Bob wants to communicate secretly. To achieve this goal, Alice will create a training set. The training set will consists of ASCII codes (decimal values) of 317 characters with their corresponding two different ciphertexts (C1 and C2). The ciphertexts C1 and C2 are of different lengths. The ciphertexts are formed by padding the decimal digits with a random key as shown in Figure 2.

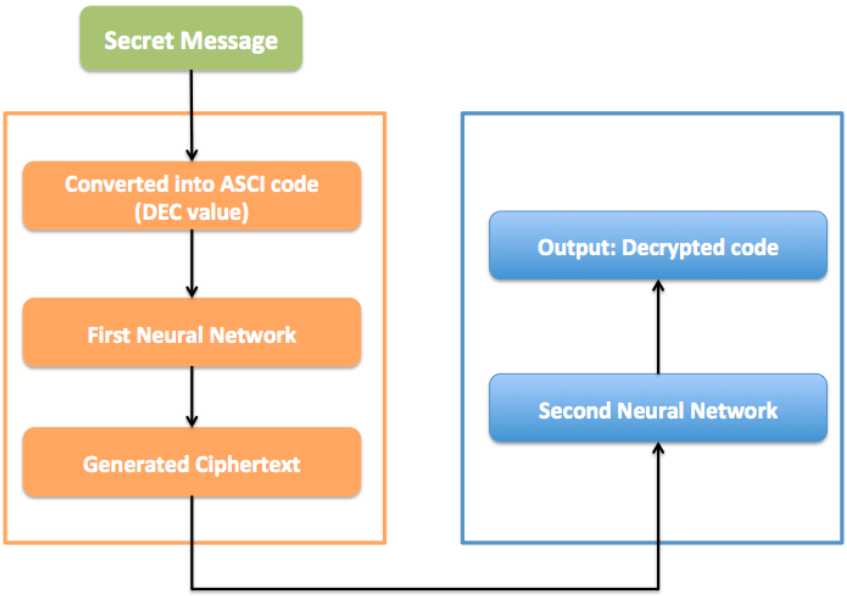

Fig. 1. Encryption Model based Artificial Neural Network

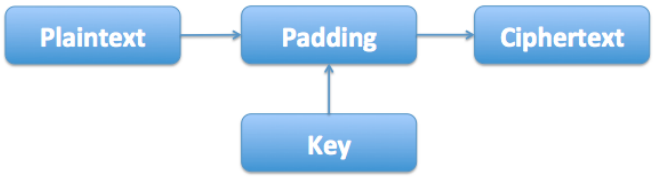

Fig. 2. Formation of Ciphertext

For instance, the ciphertext C1 is formed by taking a random key and padding the ciphertext with that key. Alice will then secretly hand over the entire training data to Bob. Both Alice and Bob will train their neural networks with their training data. Alice will train her neural network by using the ASCII/Decimal values as the feature vector and predicting the ciphertext. Bob on the other hand will train his neural network by taking the ciphertext as the feature vector and predicting the decimal value.

As Alice wants to share some secret information with Bob, she has to convert the secret information or test message to ASCII (decimal) values. The decimal values will be the input to the first Neural network. The first neural network will then produce an output or a ciphertext. Afterwards, Alice will hand over the ciphertext to Bob. Bob will use this ciphertext as an input to the second neural network and will get the decrypted text message.

As our proposed model is based on cryptology, we proved that our proposed model cannot be attacked by any other adversary [16]. For instance, suppose Eve (an adversary) wants to eavesdrop on Alice's and Bob's conversation. To do this, Eve can attack the model in two ways. First, Eve can mimic the two neural networks, however, that is impossible for Eve to do because she does not have the training set. Secondly, Eve can randomly start guessing the ciphertexts, however, in cryptography privacy is preserved against efficient adversaries that run in a feasible amount of time [17]. Thus, Eve who has 'computational power polynomial in time' would not be able to go through all keys if we use random keys. For privacy, we keep on changing the key for each sets of messages so that no adversary can guess the right key. Hence, our model is most secured to maintain confidentiality and integrity of the data.

4 Experimental Setup and Results

We divided this section into 'Experimental Setup' and 'Results'. The experimental setup is divided into the encryption setting and decryption setting. For both the settings, we included different parameters with different ciphertexts to measure the accuracy of our model. The length of the different ciphertexts C1 and C2 are of medium and large, respectively. The parameters are hidden network structures, step values, learning rates and optimizers. The results include the evaluation of the best ciphertext with the best network structure for both the encryption and the decryption setting.

4.1 Experimental Setup

In the encryption setting we considered evaluating the performance of two different ciphertexts. First, we consider evaluating the performance of C1, which is of medium length. For the evaluation we included different hidden layer network structures and different optimizers as shown in Table 1. RMSE values are root mean squared error values. Also, values displayed in bold are the best RMSE values achieved with a particular setting. The optimizer listed as 'Default' in the tables is the Adagrad optimizer.

In the next step since the RMSE value with the three layer hidden network structure and with the Adam optimizer returned the best score, we increased the step value to see how it affects the RMSE value (Table 2). We also applied the same network structure with a step value of 3000 to different learning rates (Table 3).

Then, we evaluated the ciphertext C2 with a hidden network structure of [350, 350, 350] and [350, 350] as summarized in Table 4. We took the step value as 4000 since the length of ciphertext C2 is very large.

From Table 4, we can interpret that for Ciphertext C2, the RMSE value is best when we use the two layer network structure with the Adam optimizer and a step value of 4000. Table 5 shows how different step values change the RMSE values.

Table 1. Ciphertext (C1) with different optimizers

Hidden Layers	**Optimizers**	**RMSE**
[350,350,350]	Default	6.220
[350,350,350]	**Adam**	**0.371**
[350,350,350]	Adagrad	243.101
[350,350,350]	Proximal	3.913
[350,350]	Default	0.493
[350,350]	Adam	0.455
[350,350]	Adagrad	1125.941
[350,350]	Proximal	16.672

Table 2. Ciphertext (C1) with different step values

Step values	**RMSE**
1000	0.371
2000	0.341
3000	**0.224**
4000	0.533

Table 3. Ciphertext (C1) with different learning rate

Learning rates	**RMSE**
0.010	**0.224**
0.001	0.901
0.0001	24.756

Table 4. Ciphertext (C2) with different optimizers

Hidden Layers	**Optimizers**	**RMSE**
[350,350,350]	Default	11.804
[350,350,350]	Adam	4.590
[350,350,350]	Adagrad	19.976
[350,350,350]	Proximal	18.762
[350,350]	Default	17.743
[350,350]	**Adam**	**4.519**
[350,350]	Adagrad	14471.780
[350,350]	Proximal	163.521

Table 5. Ciphertext (C2) with different step values

Step values	**RMSE**
4000	**4.590**
5000	16.935
6000	16.443
7000	14.309

Since we have seen that the learning rate plays a very important role in Ciphertext C1, we experimented with C2 with different leaning rates and with a two layer hidden network structure and the Adam optimizer (Table 6).

Table 6. Ciphertext (C2) with different learning rates

Learning rates	**RMSE**
0.01	**4.963**
0.001	8.335
0.0001	148.581

For the decrypting setting, in order to decrypt the test message, which is encrypted with the Ciphertext C1, we use different network structures with a step value of 1000. We investigated how well the neural network can decrypt the test message with different optimizers (Table 7).

Table 7. Ciphertext (C1) with different optimizers

Hidden Layers	**Optimizers**	**RMSE**
[150,150,150]	Default	0.333
[150,150,150]	Adam	19.743
[150,150,150]	Adagrad	40.671
[150,150,150]	Proximal	0.175
[150,150]	**Default**	**0.082**
[150,150]	Adam	111.951
[150,150]	Adagrad	161.564

Then, we use the two layer hidden network structure to find if different step values can give a better RMSE value. The output is given in Table 8. From the table, we can say that that the default optimizer with the two layer hidden

network we obtained the best RMSE value, so we interpret the optimizer with different learning rates shown in Table 9.

Table 8. Ciphertext (C1) with different step values

Step values	**RMSE**
1000	0.333
2000	0.139
3000	**0.112**
4000	0.144

Table 9. Ciphertext (C1) with different learning rates

Learning rates	**RMSE**
0.01	**0.175**
0.001	43.976
0.0001	187.124

For decrypting the test message with Ciphertext C2, we implemented different hidden layer network stuctures as before with a step value of 3000. The output is summarized in Table 10.

Table 10. Ciphertext (C2) with different optimizers

Hidden Layers	**Optimizers**	**RMSE**
[150,150,150]	Default	0.174
[150,150,150]	Adam	0.595
[150,150,150]	Adagrad	17.755
[150,150,150]	Proximal	0.1296
[150,150]	**Default**	**0.043**
[150,150]	Adam	6.981
[150,150]	Adagrad	111.393
[150,150]	Proximal	0.278

We implemented different step values with the two layer hidden network structure and the default optimizer (Table 11) and different learning rates with the same hidden layer network structure using the default optimizer (Table 12).

Table 11. Ciphertext (C2) with different step values

Step values	**RMSE**
3000	0.174
4000	0.232
5000	0.162
6000	**0.094**

Table 12. Ciphertext (C2) with different learning rates

Learning rates	**RMSE**
0.01	**0.130**
0.001	4.134
0.0001	183.270

4.2 Results

We evaluated both the encryption and the decryption setting separately. From Table 1-6, we found the best network structure among the two ciphertexts for encrypting a message is ciphertext C1 with a hidden layer of [350, 350, 350], learning rate 0.01, a step value of 3000 and with the Adam optimizer. From Table 7-12, we found the best network structure among all the ciphertexts that decrypt a test message is ciphertext C2 with a hidden layer structure of [150, 150], step value of 3000, learning rate of 0.01 and with the default optimizer.

5 Conclusion

In this paper, we reviewed several related research work done in the neural cryptography area. In our paper, we proposed a method for both encrypting and decrypting a ciphertext using a neural network. Our model takes decimal values as input and converts them to ciphertext, and then again to the desired output. We proved how the model is secured against brute force attacks thereby preserving privacy and security.

The experiments shows how different network structures perform with different learning rates as well as different step values and optimizers. Also as was shown with the experiments, the RMSE value of the training data played a vital role. Even though we have shown that the ciphertext which is of medium length performs best in encrypting a test message, however, we highly recommend to use ciphertexts of large length to prevent any confidentiality leak of data. If the ciphertext is large then it would be very difficult for an adversary to traverse through all the keys in polynomial time.

In the future, an asymmetric key encryption based neural network should be used to improve the integrity and confidentiality of the data. In addition, future

work should also involve finding a neural network which can do both encryption and decryption simultaneously.

References

1. Arvandi, Maryam, et al. *Symmetric cipher design using recurrent neural networks* IJCNN'06. International Joint Conference on. IEEE, 2006.
2. Rivest, Ronald L., Adi Shamir, and Leonard Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM 21.2 (1978): 120-126.
3. Monrose, Fabian, et al. *Cryptographic key generation from voice.* Security and Privacy, 2001. Proceedings. 2001 IEEE Symposium on. IEEE, 2001.
4. Pointcheval D. *Neural Networks and their Cryptographic Applications.* Pascale Charpin Ed. India, 1994.
5. Guo, Donghui, Lee-Ming Cheng, and L. L. Cheng. *A new symmetric probabilistic encryption scheme based on chaotic attractors of neural networks.* Applied Intelligence 10.1 (1999): 71-84.
6. Su, Scott, Alvin Lin, and Jui-Cheng Yen. *Design and realization of a new chaotic neural encryption/decryption network.* Circuits and Systems, 2000. IEEE APCCAS 2000. The 2000 IEEE Asia-Pacific Conference on. IEEE, 2000.
7. Kinzel, Wolfgang, and Ido Kanter. *Neural cryptography.* Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on. Vol. 3. IEEE, 2002.
8. Rosen-Zvi, Michal, et al. *Mutual learning in a tree parity machine and its application to cryptography.* Physical Review E 66.6 (2002): 066135.
9. Klimov, Alexander, Anton Mityagin, and Adi Shamir. *Analysis of neural cryptography.* International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2002.
10. Ruttor, Andreas, et al. *Neural cryptography with feedback.* Physical Review E 69.4 (2004): 046110.
11. Ruttor, Andreas, et al. *Genetic attack on neural cryptography.* Physical Review E 73.3 (2006): 036121.
12. Sarasamma, Suseela T., Qiuming A. Zhu, and Julie Huff. *Hierarchical Kohonenen net for anomaly detection in network security.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35.2 (2005): 302-312.
13. Prabakaran, N., P. Loganathan, and P. Vivekanandan. *Neural cryptography with multiple transfers functions and multiple learning rule.* International journal of soft computing 3.3 (2008): 177-181.
14. Volna, Eva, et al. *Cryptography Based On Neural Network.* ECMS. 2012.
15. Hecht-Nielsen, Robert. *Theory of the backpropagation neural network.* Neural Networks 1.Supplement-1 (1988): 445-448.
16. Stallings, William, and Mohit P. Tahiliani. *Cryptography and network security: principles and practice.* Vol. 6. London: Pearson, 2014.
17. Micali, Silvio, and Leonid Reyzin. *Physically observable cryptography.* Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2004.