ADDRESSING CHALLENGES IN DATA PRIVACY AND SECURITY: VARIOUS APPROACHES TO SECURE DATA

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By Sayantica Pattanayak

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Computer Science

May 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ADDRESSING CHALLENGES IN DATA PRIVACY AND SECURITY:
VARIOUS APPROACHES TO SECURE DATA

By

Sayantica Pattanayak

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Simone A. Ludwig
Chair

Dr. Anne Denton

Dr. Pratap Kotala

Dr. Dogan Comez

Approved:

August 21, 2021	Dr. Simone A. Ludwig
Date	Department Chair

ABSTRACT

Emerging neural networks based machine learning techniques such as deep learning and its variants have shown tremendous potential in many application domains. However, the neural network models raise serious privacy concerns due to the risk of leakage of highly privacy-sensitive data. In this dissertation, we propose various techniques to hide the sensitive information and also evaluate the performance and efficacy of our proposed models.

In our first research work we propose a model, which can both encrypt and decrypt a ciphertext. Our model is based on symmetric key encryption and back propagation neural network. Our model takes the decimal values and converts them to ciphertext and then again to decimal values.

In our second research work, we propose a remote password authentication scheme using neural network. In this model, we have shown how an user can communicate securely with more than one server. A user registers himself / herself with a trusted authority and gets a user id and a password. The user uses the password and the user id to login to one or multiple servers. The servers can validate the legitimacy of the user. Our experiments use different classifiers to evaluate the accuracy and the efficiency of our proposed model.

In our third research paper, we develop a technique to securely send patient information to differ- ent organizations. Our technique used different fuzzy membership functions to hide the sensitive information about patients.

In our fourth research paper, we introduced an approach to substitute the sensitive attributes with the non-sensitive attributes. We divide the data set into three different subsets: desired, sensitive and non-sensitive subsets. The output of the denoising autoencoder will only be the desired and non-sensitive subsets. The sensitive subsets are hidden by the non-sensitive subsets. We evaluate the efficacy of our predictive model using three different flavors of autoencoders. We measure the F1-score of our model against each of the three autoencoders. As our predictive model is based on privacy, we have also used a Generative Adversarial Neural Network (GAN), which is used to show to what extend our model is secure.

ACKNOWLEDGEMENTS

First and foremost I wish to thank my academic advisor Dr. Simone A. Ludwig for her constant support during my PHD program. Dr Ludwig has given me the freedom to pursue various research without any objections. She also provided insightfull discusions about the research. Dr Ludwig was not only my academic advisor but a mentor to me. The passion and commitments she put into her students are highly appreciable .

I would like to extend my sincere thanks to Dr Pratap Kotala, Dr Anne Denton and Dr Dogan Comez for serving on my graduate committee. They dedicated their valuable time to provide feedback on my thesis and helping me in professional developments.

I would also like to thank Guy Hokanson, the system administrator of the department. His constant help through emails and in-person in different technical issues has helped me to finish my thesis in time. Also I would like to thank my lab members for their support.

My acknowledgement does not end without thanking my husband and my kids. Their constant sacrifices, support and patience has inspired me to finish my defense. My only wish was my Dad was around me to see me earn the degree. He would have been proud of me.

DEDICATION

To my husband and my kids

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The value of collecting data has skyrocketed ever since Glaxo Smith Kline got access to genetic data from the genomics and biotechnology company 23andMe [1]. The goal of the part-nership was to gather insights and discover novel drug targets driving disease progression, and to develop therapies for serious medical needs based on those discoveries. However, this lead to serious privacy concerns among customers. Privacy Concern issues arose after several breathtaking series of scandals. Cambridge Analytica [2] had collected and exploited Facebook user data. Snowden [3] leaked highly classified documents of the National Security Agency. Also millions of unique passport numbers had been stolen in a Marriott International data breach [4]. The consequences of these scandals lead to an extensive rewrite of privacy law in Europe [5]. As a result, companies tend to be incredibly protective of their data assets and very conservative when it comes to share them with other companies. At the same time, data scientists need access to large labeled data sets in order to validate their models. This constant friction between privacy and learning has become a governing dynamic in modern machine learning applications.

For any company, it will be immensely valuable to give access to their data to a large group of data scientists and researchers so that they can extract information from it, compare and evaluate mod- els, and arrive at a satisfactory solution. However, how can companies guarantee that their data will be protected or that the interest of their customers will be preserved or that some unscrupulous data scientists would not share their intelligence with the competition? When confronted with that issue, many companies started anonymizing their data sets but that is hardly a solution in many scenarios.

The following sections briefly describe the various privacy methods adapted. A brief description of the background is presented in Sections 1.1-1.3. The motivation of the work is discussed in Section 1.4. The contributions of the work is described in Section 1.5, and an overview of the dissertation is listed in Section 1.6.

1.1. Cryptography

Cryptography is a method of protecting information and communication through the use of codes so that only those for whom the information is intended can read and process it. The pre-fix “crypt” means “hidden” or “vault”, and the suffix “graphy” stands for “writing”.

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts, and a set of rule-based calculations called algorithms to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation and digital signing and verification to protect data privacy, web browsing on the internet, and confidential communications such as credit card transactions and email.

1.1.1. Cryptography Techniques

Cryptography is closely related to the disciplines of cryptology and cryptanalysis. It includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today’s computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as cleartext) into ciphertext (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers. Modern cryptography concerns itself with the following four objectives:

- Confidentiality: the information cannot be understood by anyone for whom it was unintended.
- Integrity: the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected.
- Non-repudiation: the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information.
- Authentication: the sender and receiver can confirm each others identity and the origin/destination of the information.

1.2. Cryptographic Algorithms

Cryptosystems use a set of procedures known as cryptographic algorithms, or ciphers, to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications.

- Symmetric Key Algorithm

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

$$C = Enc_k(m) \tag{1.1}$$

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

$$m = Dec_k(C) \tag{1.2}$$

- Asymmetric Key Algorithm

Public-key encryption or Asymmetric 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 [6], 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.

1.3. Homomorphic Encryption

Homomorphic Encryptions is another form of hiding data. Homomorphic Encryptions rep- resents one of the biggest breakthroughs in the cryptography space. This technique is likely to become the foundation of many AI applications in the near future. Conceptually, the term Homomorphic Encryption describes a class of encryption algorithms which satisfy the Homomorphic property: that is certain operations, such as addition, can be carried out on cipher texts directly so that upon decryption the same answer is obtained as operating on the original messages. In the context of AI, Homomorphic Encryption could enable data scientists to perform operations on encrypted data that will yield the same results as if they were operating on clear data. The main issue with Homomorphic Encryption is that the technology is still too computationally expensive to be considered in mainstream applications. Basic Homomorphic Encryption techniques can turn 1 MB of data into 16 GB, which makes it completely impractical form many AI scenarios.

1.4. Neural Cryptography

Somewhere between Anonymization methods and Homomorphic Encryption, a novel tech- nique pioneered by Google, uses adversarial neural networks to protect information from other neural network models. The technique is known as Neural Cryptography.

Traditionally, neural networks have been considered to be very bad at cryptographic operations as they have a hard time performing a simple XOR computation. While that is true, it turns out that neural networks can learn to protect the confidentiality of their data from other neural networks. They discover forms of encryption and decryption, without being taught specifically for these purposes. In short, Neural Cryptography is a branch of cryptography, which is the most significant part of communication security [7].

1.5. Artificial Neural Network

Neural Cryptography uses Artificial Neural Network models to achieve privacy and security. Artificial Neural Networks are computational models which work similar to the functioning of a human nervous system. There are several kinds of artificial neural networks. These type of networks are implemented based on the mathematical operations, and a set of parameters required to determine the output. Different Neural Network models, which have been used in this dissertation are described below:

1.5.1. Perceptron

This is the most simplest and the oldest form of Neural Network. The neuron takes the inputs sum them up, applies some activation function, and passes the result to the output layer, which consists of one neuron [8].

1.5.2. Feed Forward Neural Network (FF)

This type of Neural Network [9] model was originated in 1950. The Neural Network model has more than one neuron. The model has at least one hidden layer between the input and the output. All the inputs are connected to each of the nodes in the hidden layer. The activation flows from input to output without back loops. In a Feed Forward Neural Network, the sum of the products of the inputs and their weights are calculated. This is then fed to the output. Feed forward neural networks are used for technologies such as face recognition and computer vision.

1.5.3. Radial Basis Neural Network (RBF)

RBF uses the radial basis function as the activation function. This type of Neural Network model works very well for classification and decision making. However, RBF do not work so well with continuous values. A radial basis function considers the distance of any data point relative to the center. Such Neural Networks have two layers; in the inner layer, the features are combined with the radial basis function [10]. The radial basis function neural network is applied extensively areas such as power restoration systems [11].

1.5.4. Deep Feed Forward Neural Network (DFF)

DFF is now the most applied of the different machine learning systems [12]. The network uses more than one hidden layer. DFF covers the same purpose as FF. However, their performance is much better than FF. DFF is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer. This type of Neural Network is applied extensively in speech recognition and machine translation technologies.

1.5.5. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) uses a variation of multilayer perceptrons. A CNN contains one or more than one convolutional layer(s). These layers can either be completely interconnected or pooled. Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters. Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems [70].

1.5.6. Long Term Short Memory Neural Network (LSTM)

A LSTM Neural Network is a type of Artificial Neural Network in which the output of a particular layer is saved and fed back to the input. This helps predict the outcome of the layer. From each time-step to the next, each node will remember some information that it had in the previous time-step. In other words, each node acts as a memory cell while computing and carrying out operations. LSTM begins with the front propagation as usual but remembers the information it may need to use later. This type of Neural Network is very effective in text-to-speech conversion technology and time series analysis [14].

1.5.7. Autoencoder Neural Network (AE)

Autoencoder Neural Network [67] can be trained without supervision. Their number of hidden cells is smaller than the number of input cells (and the number of output cells equals the number of input cells), and when the AE is trained the way the output is as close to the input as possible, forces AEs to generalize data and search for common patterns. Autoencoders are used for classification, clustering, and feature compression.

1.6. Motivation and Problem Definition

Although everyone understands the concept of privacy, there is no universally accepted definition of privacy. Privacy can be defined in three ways. Privacy in Information deals with the handling and collection of private data. Communication Privacy deals with privacy while communicating. Territorial Privacy is concerned with the invasion of physical boundaries. With the latest advancement in computer science, privacy in information and communication is getting more attention. Many organization share information and messages to other organization for analyzing the data. However, sharing raw information or messages is a potential threat to privacy.

Confidentiality in communication or messages can be achieved using Neural Cryptography 1.3. Neural Cryptography maintains 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 4.6 provides secret communication between two parties either by symmetric key encryption or asymmetric key encryption. In symmetric key encryption, both parties shares the same key. However, in asymmetric key encryption both parties share different keys.

On the other hand privacy in information can be achieved by sanitizing the sensitive information. When raw data is shared there is a chance of a privacy breach. For instance, banks might wish to collaborate in order to detect the fraudulent behavior of customers. This requires the bank to share financial records of the customer. Also, hospitals want to share data of patients with the other hospitals for efficient diagnosis of diseases. In both cases, the bank and the hospitals hold shared data without violating the privacy of the individual customer.

The motivation are summarized below:

Cryptography has been used as a tool for sending confidential data from one Neural Network to another Neural Network. The two Neural Networks named Alice and Bob trained their respective data sets. Alice trained the model using her encryption data set, and Bob trained his model using his decryption data set. As Neural Networks learn from training, they can now encrypt and de- crypt any confidential data sets. The two neural network share a same secret key to secure their communication. Eve, an eves dropper, will not be able to learn the private message they shared. Eve can only make a guess to the message.

Cryptography is also used as a tool in remote password authentication scheme for multiple server. Here, a neural network can login into multiple servers using a single password. The authentication scheme is made secure using Diffie Hellman Key exchange protocol [17]. The key is shared with the user, server and the trusted party. The neural network is based on supervised learning and back propagation Neural network.

Privacy is needed not only in communication. Privacy plays a very vital role when sending in- formation. Nowadays many organizations share data. The sharing of data helps to analyze the data for future research work. However, if the data contains personal information, then the privacy is compromised. Fuzzy membership functions are use to sanitized the personal information. The personal information are hidden with the boundary values of different fuzzy membership functions. The sanitized data set is sent to other organization using Autoencoder neural network. The Autoencoder neural network is used in compressing the attributes to most important ones.

Analyzing the efficacy of the neural network model is very crucial. To improve the efficiency we used different techniques. The various techniques are described below.

1.6.1. Overfitting

The first step in ensuring our neural network performs well on the testing data is to verify that our neural network does not overfit. Overfitting happens when the model starts to memorize values from the training data instead of learning from them. Overfitting can be identified by observing training and testing accuracy. If the training accuracy is higher than the testing accuracy then it is assumed that the model has overfitted. To avoid overfitting we used following techniques:

- Regularization of data (L1 or L2): This technique reduces overfitting by adding a penalty to the loss function [15].
- Early Stopping: This can be done by stopping the training before the weights have converged [16].
- Dropout: At each training phase, individual neurons are dropped out of the network [15].

1.6.2. Hyper-parameter Tuning

Each neural network will have its best set of hyperparameters, which will lead to the maxi- mum accuracy. Unfortunately, there is no direct method to identify the best set of hyperparameter for each neural network, so it is mostly obtained through trial and error. The hyperparameters which we used are the following:

- Learning Rate
- Optimizer and Loss Functions
- Batch Size and Number of Epochs
- Activation Functions

1.6.3. Resampling of Data

After performing all the above mentioned techniques, if our model did not obtain better accuracy, we resampled the data set. Resampling the data set can be done by balancing an unbalanced data sets. Also, by removing the columns and rows which have most missing values.

1.7. Contributions

This dissertation makes several contributions towards data privacy using cryptography and Neural Networks. The first chapter proposes a model which can secure privacy in communication. Two Neural Networks communicate securely in presence of a third Neural Network although the third Neural Network will not have any information of their communication. The dissertation also proposes a technique where one Neural Network can communicate securely with more than one Neural Network. The proposed technique is achieved through a remote password authentication method. Using this technique an user can log into one or multiple servers securely. Privacy plays a very vital role not only in communication but also in sharing information. Hence in this dissertation we also propose an approach of hiding the sensitive attributes of patients. Our technique can not only hide the sensitive attributes but also send the attributes to other organizations securely. This technique could help in sharing patients' data without compromising their privacy. Finally, LSTM and denoising neural network is proposed to replace the sensitive information with non-sensitive information. The contributions can be summarized as follows:

A method for both encrypting and decrypting a ciphertext using a neural network is proposed. The model takes decimal values as input and converts them to ciphertext, and then again to the desired output. The model is secured against brute force attacks thereby preserving privacy and security. The model is experimented with ciphertexts of different length and with different other parameters like network structure, learning rates, optimizers and step values. Details of the proposed method is discussed in Chapter 2.

A password authentication scheme based on Neural Networks is developed. The system identify legitimate users in real time using a pattern classification technique. The scheme is applicable to multiserver network architectures. The new scheme also shows how to securely establish a shared secret key, which is the key that will be used for encrypting/decrypting the password. Details of the proposed model is discussed in Chapter 3.

In Chapter 4, the proposed model uses fuzzy membership to hide the sensitive attributes. By hiding the data using different fuzzy membership functions, it will be difficult for some other party to identify the patient. Furthermore, to send data to different organizations an Autoencoder neural network was used. The Autoencoder retrieves the original data as input.

In chapter 5, the proposed approach shows how a predictive model substitutes the sensitive at- tributes of a data set with non-sensitive attributes. The time series data sets are divided into three information sets: desired, sensitive and non-sensitive. The desired information will be used by other organizations for analysis. The sensitive information in the data sets is replaced by non-sensitive information. We used a deep autoencoder, which extracts the sensitive information and replaces those with non-sensitive information.

1.8. Dissertation Overview

This dissertation is a paper-based version, where each chapter has been derived from the papers published during the PhD work. This is an overview of the remaining chapters of this dissertation.

In Chapter 2, a method for encrypting and decrypting ciphertext is proposed. The chapter is derived from the following publication:

- Sayantica Pattanayak and Simone A. Ludwig. "Encryption based on Neural Cryptography." In International Conference on Health Information Science (pp. 321-330), New Delhi, India, December 2017.

In Chapter 3, a remote access authentication method is proposed. The chapter is derived from the following publication:

- Sayantica Pattanayak and Simone A. Ludwig. "A Secure Access Authentication Scheme for Multiserver Environments Using Neural Cryptography", Journal Of Information Assurance And Security , 13(1), (2018),56-65.

In Chapter 4, a technique was developed not to hide the data but to send the data to other organizations securely. The chapter is derived from the following publication:

- Sayantica Pattanayak and Simone A. Ludwig. "Improving Data Privacy using Fuzzy Logic and Autoencoder Neural Network", 2019 IEEE International Conference on Fuzzy Systems, New Orleans, USA, June 2019.

In Chapter 5, a technique was developed to substitute sensitive attributes with non-sensitive attributes. The chapter is derived from the following publication:

- Sayantica Pattanayak and Simone A. Ludwig. " Analyzing Privacy of Time Series Data using Substitute Autoencoder Neural Network", IEEE Symposium on Computational Intelligence (SSCI 2020), Cranberra, Australia, December 2020.

2. ENCRYPTION BASED ON NEURAL CRYPTOGRAPHY

Neural Network Cryptography is an interesting area of research in the field of computer science. This chapter proposes a new model to encrypt/decrypt a secret code using Neural Networks, which is an interesting potential source of private key cryptography model that is not based on number theoretic functions. In the first part of the chapter we proposed our model as well as 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 chapter, we experiment with the neural network model using three 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.

2.1. 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 [18]. 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 [19]. The authors showed how a discrete Hopfield Neural Network (HNN) model is favorable 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). In the case of overstorage, i.e., the Overstoraged HNN (OHNN), the number of stored patterns will be larger than that of a HNN and the system initial state will then converge to one of the system attractors compared to conventional HNN [20]. Conventional

HNN used a property of HNN to randomize the outcome of an event which can eventually be used for encryption schemes.

Another chaotic neural network and its VLSI architecture for digital signal encryption and decryption was proposed in 2000 [80]. Chaotic neural network are used for encryption as well as for decryption because of high security, no distortion in signal and also suitability of system integration. In [75], the authors demonstrate that neural networks can learn to protect communications. The learning does not require prescribing a particular set of cryptographic algorithms, nor indicating ways of applying these algorithms. The learning is based only on a secrecy specification represented by the training objectives. Precisely, the paper relates the application of neural network to multiagent tasks.

Another paper proposed the basis of the Neural Key Exchange protocol [23]. Neural Key Exchange Protocol is based on the synchronization of the weights of a Tree Parity Machine (TPM) [24]. 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 [25]. The attacks are Geometric attack [26], Genetic attack [27], and probabilistic analysis [28]. These attacks can be fixed by addition of a feedback mechanism [29].

Authors in [30] 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.

2.2. Proposed Approach

Our neural network model is based on Backpropagation network [31]. A Backpropagation network is one of the most complex neural networks used for supervised learning. The algorithm assumes a feedforward NN architecture.

Our proposed model (Fig 2.1) is based on symmetric key encryption. Like in cryptography we have Alice and Bob, here we will assume that Alice and Bob want to communicate secretly. Alice and Bob share their secret keys with each other using symmetric key encryption. Then, Alice shares her 317 characters with Bob. Alice and Bob create their own training sets. To achieve this goal, Alice will create a training set as she has the secret keys. The training set will consists of ASCII codes (decimal values) of 317 characters with their corresponding three different ciphertexts (C1, C2 and C3). The ciphertexts C1, C2 and C3 are of different lengths.

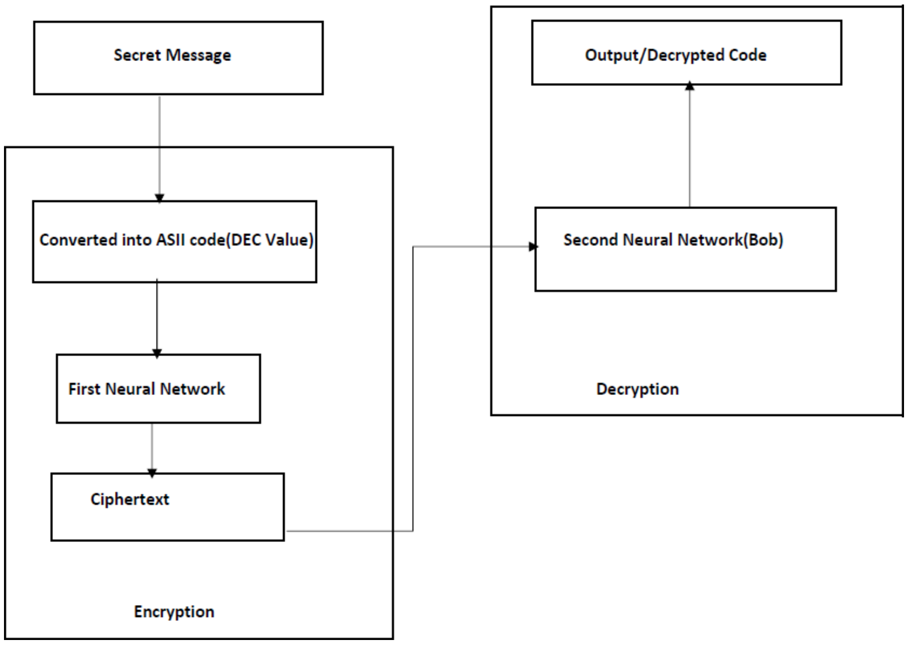

Figure 2.1. Encryption model based artificial neural network.

The ciphertext are formed by padding the decimal digits with secret keys as shown in Figure 2.2. Bob will create his own training set with three different ciphertexts (C1, C2 and C3) as feature vectors and predicting the ASCII values of the 317 characters.

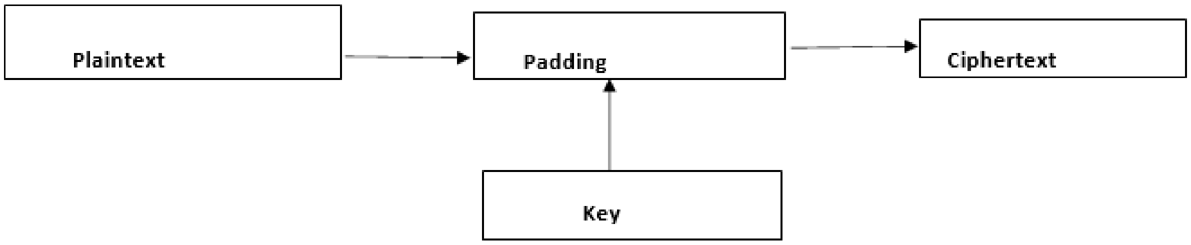

Figure 2.2. Formation of ciphertext

For instance, the ciphertext C1 is formed by taking a secret key and padding the ciphertext with that key. The secret key is randomly generated. Alice will then share the secret key with Bob. Alice and Bob will share the 317 characters with each other. Alice and Bob will create their own training set as they have the secret key. 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 the 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 [32]. For instance, suppose Eve (an adversary) wants to eavesdrop to 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 he 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 [33]. 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.

2.3. Experimental Setup and Results

We divided this section into experimental setup and findings. The experimental set is divided into Encryption setting and Decryption Setting. For both the settings we included different parameters with different cipher text to measure the accuracy of our model. The length of different ciphertext C1, C2 and C3 are medium, small and large respectively. The parameters are hidden network structures, step values, learning rates and optimizers. The result includes the evaluation of the best ciphertext with the best network structure for both encryption and decryption.

2.3.1. Experimental Setup

In the encryption setting we considered evaluating the performance of three different ciphertexts. First we consider evaluating the performance of C1, which is of medium length. For evaluation we included different hidden layers network structure and different optimizers Table 2.1.

Table 2.1. Ciphertext (C1) with different optimizers

Hidden Layers	Optimizers	RMSE
[350,350,350]	Default	6.22
[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
[350]	Default	47.806
[350]	Adam	3.609
[350]	Adagrad	1224.041
[350]	Proximal	1044.983

In the next step as the RMSE value with three layer hidden network structure and with Adam optimizer gave us a good score, we increased the step value to see how it effects the RMSE value as shown in Table 2.2. We also applied the same network structure with step value of 3,000 to different learning rates as shown in Table 2.3.

Table 2.2. Ciphertext (C1) with step values

Step values	RMSE
1000	0.371
2000	0.341
3000	0.224
4000	0.533

Now for ciphertext C2 which is of small length, we experimented with three different hidden network structure. First we took three layers [350, 350, 350], then [350, 350] and [350] keeping the step value to 1000. The results are summarized in Table 2.4.

Table 2.3. Ciphertext (C1) with learning rate

Learning Rates	RMSE
0.010	0.224
0.001	0.901
0.0001	24.756

Table 2.4. Ciphertext (C2) with different optimizers

Hidden Layers	Optimizers	RMSE
[350,350,350]	Default	0.741
[350,350,350]	Adam	0.028
[350,350,350]	Adagrad	29.342
[350,350,350]	Proximal	0.261
[350,350]	Default	0.091
[350,350]	Adam	0.037
[350,350]	Adagrad	148.662
[350,350]	Proximal	0.926
[350]	Default	1.4333
[350]	Adam	0.272
[350]	Adagrad	229.961
[350]	Proximal	64.143

Now as we see that the Ciphertext C2 gives a best RMSE value with two layer [350,350] hidden network structure and Adam optimizer. So we evaluated this hidden layer structure with different step values in 2.5. Now we evaluated the learning rates with two layer hidden network structure 2.6. We kept the step value to 4000 and used Adam optimizer as this gave a good RMSE

Table 2.5. Ciphertext (C2) with step values

Step values	RMSE
1000	0.091
2000	0.064
3000	0.202
4000	0.033

Table 2.6. Ciphertext (C2) with learning rates

Learning Rates	RMSE
0.010	0.037
0.001	0.931
0.0001	27.721

We evaluated the ciphertext C3 with hidden network structure [350, 350, 350], [350, 350], [350] and summarized in Table 2.3.1. Here we took the step value as 4000 as the length ciphertext C3 is very large.

Table 2.7. Ciphertext (C3) 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
[350]	Default	3231.150
[350]	Adam	15.045
[350]	Adagrad	15155.403

From the Table we can interpret that for the Ciphertext C3 RMSE value is best when we use three layer network structure with Adam optimizer and step value of 4000. Table 2.8 shows how different step values changes the RMSE values. As we have seen learning rates play a very important role in Ciphertexts C1 and C2. We experimented C3 with different leaning rates and with three layer hidden network structure and Adam optimizer Table 2.9.

Table 2.8. Ciphertext (C3) with step values

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

Table 2.9. Ciphertext (C3) with learning rates

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

In the Decrypting setting, to decrypt the test message which is encrypted with the Cipher- text C1 we use different network structure with step value of 1000 and proximal gradient descent optimizer. We tried to find how good the neural network can decrypt the test message with different optimizer Table 2.10.

Now we use the three layer hidden network structure and try to find if different step values can give us a better RMSE value. The output is in Table 2.11. From the table we can say that that proximal gradient optimizer with three layer hidden network and step values we got a best RMSE value, so we interpret the optimizer with different learning rates Table 2.12 For decrypting the message encrypted using the Ciphertext C2 we use three different network structure. The hidden layer network structure are [150, 150, 150], [150, 150], [150] with step value of 1000. The output is in Table 2.13.

Here also we interpret the different hidden layer network structure of C2 and implemented two layer network structure with proximal gradient descent optimizer to different step values Table

2.14. Also we implemented the three layer hidden network structure with proximal gradient optimizer and step value 1000 to different learning rates Table 2.15.

Table 2.10. 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
[150,150]	Proximal	1.721
[150]	Default	1.754
[150]	Adam	185.184
[150]	Adagrad	186.592
[150]	Proximal	93.032

Table 2.11. Ciphertext (C1) with step values

Step values	RMSE
1000	0.333
2000	0.139
3000	0.1120
4000	0.144

Table 2.12. 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 C3, we implemented three different hidden layer network structure as before with step value of 3000 and the output is summarized in Table 2.16.We implemented different step values with three layer hidden network structure and proximal gradient descent optimizer Table 2.17 and different learning rates with the same hidden layer network structure and proximal gradient optimizer Table 2.18.

2.4. Results

Table 2.13. Ciphertext (C2) with different optimizers

Hidden Layers	Optimizers	RMSE
[150,150,150]	Default	0.168
[150,150,150]	Adam	21.271
[150,150,150]	Adagrad	53.487
[150,150,150]	Proximal	0.170
[150,150]	Default	0.110
[150,150]	Adam	101.217
[150,150]	Adagrad	168.567
[150,150]	Proximal	1.872
[150]	Default	0.110
[150]	Adam	101.217
[150]	Adagrad	168.567
[150]	Proximal	1.872

Table 2.14. Ciphertext (C2) with step values

Step values	RMSE
1000	0.115
2000	0.099
3000	0.087
4000	0.086

We evaluated both the encryption and the decryption setting separately. From Table 2.1, 2.2 and 2.3 we found that for encryption setting the best network structure to use, when we are encrypting the test message with Ciphertext C1 will be the three layer hidden network structure, with Adam Optimizer and learning rate as 0.01. For Encrypting with Ciphertext C2 the best network structure to use is the two layer network structure with Adam optimizer and learning rate of 0.01 (Table 2.4, 2.5, 2.6). For both of them step value was 1000. For encrypting with ciphertext C3, the best network structure will be the three layer network structure with Adam optimizer and step value of 3000 (Table 2.7, 2.8, 2.9). The best network structure among all the ciphertext for encrypting a test message is with ciphertext C2 with hidden layer [350, 350], learning rate 0.01 and step value of 1000. Fig 2.3 shows the plot of the behavior of the ciphertext C2 if we split the training data into training and test of size 0.2. However to maintain confidentiality and authenticity we should always try to use a very long key, even if the ciphertext C3 does not give us a good RMSE value.

Table 2.15. Ciphertext (C2) with different learning rates

Learning Rates	**RMSE**
0.01	0.170
0.001	38.341
0.0001	187.099

Table 2.16. Ciphertext (C3) 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
[150]	Default	0.260
[150]	Adam	166.589
[150]	Adagrad	184.544
[150]	Proximal	15.242

Table 2.17. Ciphertext (C3) with step values

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

Table 2.18. Ciphertext (C3) with different learning rates

Learning Rates	**RMSE**
0.01	0.1296
0.001	4.134
0.0001	183.270

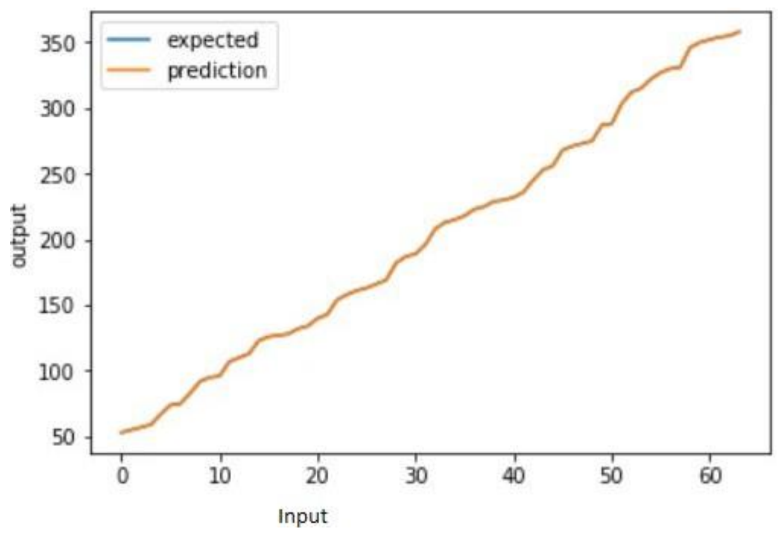

Figure 2.3. Expected and predicted value of ciphertext C2.

From Table 2.10, 2.11, 2.12, 2.13, 2.14, 2.15 we found that decrypting with ciphertext C1 and C2, two layer hidden network structure with proximal descent optimizer is the best neural network to use. From Table 2.16, 2.17, 2.18 we found that to decrypt a message with ciphertext C3 the three layer hidden network structure with proximal gradient descent optimizer is the best one. The best network structure among all the ciphertext that decrypt a test message is with ciphertext C3 with hidden layer [150, 150] and step value of 3000. Fig 2.4 shows the plot of the behavior of the ciphertext C3 when we split the training data into training and test of size 0.2.

2.5. Summary

In this chapter we reviewed the several work done in neural cryptography. In our dissertation we proposed a method for both encrypting and decrypting a ciphertext. Our model takes decimal values input and convert them to ciphertext and then again to the desired output. We proved how are model is secured against Brute Force attack. Thereby preserving the privacy and security.

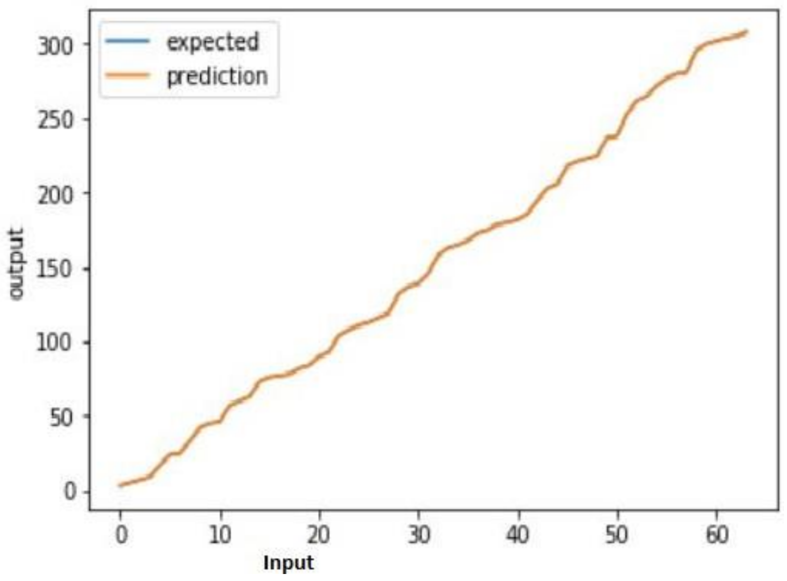

Figure 2.4. Expected and predicted value of ciphertext C3.

The experiments shows that how different network structure performs with different learning rates as well as step values and optimizers. Also the RMSE value of the training data played a vital role. Though we have shown the ciphertext which is of small length perform the best in encrypting and decrypting the test message. We highly recommend to use ciphertext of large length to prevent confidentiality of data. If the ciphertext is large than it would be very difficult for the adversary to traverse through all the keys in polynomial time.

3. A SECURE ACCESS AUTHENTICATION SCHEME FOR MULTISERVER ENVIRONMENTS USING NEURAL CRYPTOGRAPHY

With the rapid growth of network technologies, remote servers provide resources to be accessed over an open network around the world. Mainly due to the convenience of the Internet, distant users can share information with each other. In a distributed environment, secure communication in insecure communication networks is a very important issue that needs to be addressed. Hence, user authentication and secret key distribution become the most important security services for communication networks. A common feature of conventional password authentication schemes is that they use a verification table. The verification table consists of users identities and the encrypted passwords. The verification table is securely stored by the server. If the verification table is stolen or modified by an adversary, the system will be breached. Also, the conventional password authentication system is applicable to a single server. In this chapter, we designed a re- mote password authentication scheme for a multiserver environment. Our remote password based authentication method is based on artificial neural networks. In the first part of the chapter we will show how users will communicate with different servers securely. In the second part of the chapter we experimented with 540 passwords applying different classifiers. The experimental results show how the accuracy of our model can be even further improved. Furthermore, our proposed model is a more efficient and accurate authentication scheme for multiserver environments compared to other models.

3.1. Related Work

In this section, related work in the area of remote password based authentication schemes are introduced. The authors in [34] describe an efficient and secure authentication scheme for multi- server environments. Their scheme uses a hashing function to implement the mutual verification and session key agreement. The scheme does not manage the secret key table of the users and yet achieves users' anonymity. Their scheme is also nounce-based to avoid time synchronization problems. This protocol uses only a cryptographic one-way hash function for the implementation.

In the same year, Hsiang and Shih [35] found that Liao and Wangs' protocol [34] is susceptible to insider attack, masquerade attack, server spoofing attack, and registration center spoofing attack, and does not provide mutual authentication. To overcome these drawbacks, the authors proposed an improved scheme over Liao and Wangs' scheme [34]. Then in 2010, the authors in [36] showed that Hsiang and Shihs' scheme [35] is insecure against the replay attack, impersonation attack and stolen smart card attack.

To overcome these problems Amin [37] proposed an efficient dynamic ID-based remote user password authentication scheme for multi-server environments. Amin claimed that his scheme could resist off-line identity guessing attack, off-line password guessing attack, privileged insider attack, user impersonation attack, many logged-in users' attack, smart card stolen attack, and session key recovery attack. However, the authors in [38] showed that the scheme proposed by Amin is vulnerable to off-line identity guessing and off-line password guessing with the smart card stolen attack.

Going back to 2003, the authors in [39] proposed a multi-server authentication protocol based on the ElGamal digital signature scheme [40] that uses simple geometric properties of the Euclidean and discrete logarithm problem concept. The server does not require to keep any verification table but the use of public keys makes this protocol computation intensive. In that same year, the authors in [41] proposed two multi-server password authentication protocols in which the user has to communicate in parallel with all authentication servers. They proved that these protocols are provably secure in the standard model. The attacker has to compromise a minimum threshold number of servers to gain any meaningful information regarding the password of a user. These two protocols differ in the way the client interacts with the different servers. However, in these schemes, the servers are equally exposed to the user as well as to the attacker. The authors in [42] proposed a password based two-server authentication protocol in which only one server was exposed to the users. The use of public keys makes this system computationally intensive. More- over, it uses Secure Socket Layer (SSL) to establish a session key between a user and the front-end server to provide authentication but it provides only unilateral authentication.

So far we reviewed the work in password authentication based on neural networks. In 1994, the password authentication scheme based on a neural networks was initially proposed in [43]. This paper presents a new multilayer neural networks approach to identify computer users. The input vectors were made up of the time intervals between successive keystrokes created by users while typing a known sequence of characters. In 1997, the same authors [43] presented their work as a continuation of their previous work. In their previous paper only interkey times was used as feature vectors. Here, the authors used key hold time as features vectors. They found that hold times were more efficient than inter key times and the best identification performance was achieved by using both time measurements. However, both the schemes cannot withstand the replay attack [44].

Another password authentication approach based on neural networks was proposed in [45]. In this approach, a neural network is trained with the back-propagation (BP) algorithm to store the user IDs and the corresponding encrypted passwords. In this method, the system stores the weights of the trained neural network instead of the verification table [46]. As a result, the security of the system is increased. However, the scheme is not applicable for multiserver environments. In [39], the authors came up with a password authentication system based on neural networks. The scheme is applicable to multiserver environments. The authors used three types of neural network models to evaluate their performance. They used the Diffie Hellman key exchange protocol to send the password from a user to the Trusted Authority. However, the scheme is vulnerable to the man-in-the-middle attack. To overcome this drawback our proposed scheme uses the three party Diffie Hellman Key exchange protocol. Hence, our scheme is not susceptible to the man-in-the- middle-attack.

3.2. Our Proposed Approach

We propose a remote password authentication scheme using Neural Networks. This scheme is designed for multiserver environments. In our previous chapter [87], we have shown how Alice can communicate with only Bob secretly. In this chapter, we developed a scheme in which Al- ice can communicate with multiple servers. To do that she has to register herself with a Central Authority. The Central Authority will give an access key to Alice to securely communicate with multiple servers. Fig. 3.1 shows how Alice is communicating with different servers.

Our scheme has three participants. The participants are Trusted Authority (TA), Users, Servers. The entire process is divided into three phases: 1) Registration phase, 2) Login phase, and 3) Validation phase.

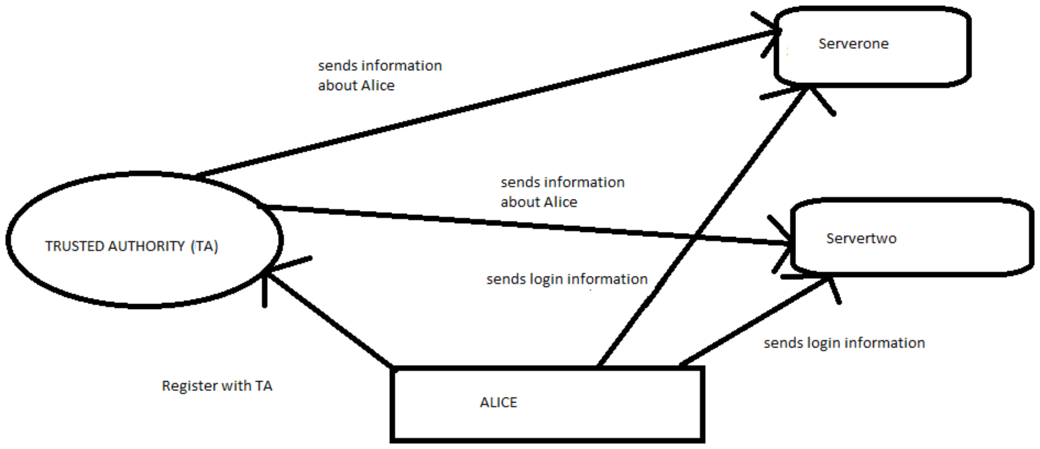

Figure 3.1. Alice communication with two different servers

3.2.1. Registration Phase

In the registration phase the new user first registers with the Trusted Authority (TA) by sharing some legitimate information. The user is granted registration for certain servers. The steps of the registration phase are as follows:

- Using the Diffie Hellman Key exchange protocol, the Users, TA and the serviceable servers create a shared secret key. The User (*U_i*) computes

$$ID_i = E_k\ (pw)$$

and sends *ID_i* to the TA. Here *pw* is the password created by the user. Without the key *k* no one can compute *ID_i*.

- In this step, the TA computes the password of user *i* as (*pw*) using

$$pw = D_k\ (ID_i)$$

Then, TA creates a training pattern using the password of the user *i*. TA adds the training pattern of the new user to create a neural network. The input units of the training pattern are the password characters. The password characters consist of English letters and/or numerals. Each password is made up of eight characters. Then, each of the password characters are mapped into a value ranging from 0 to 62 according to a mapping table. The mapping table consists of characters like lowercase letters, uppercase letters and numerals from 0 to 9.

Each of these characters are assigned a number from 0 to 62. After mapping the characters, the training set for the neural network is created. The training set consists of an output too. The output of the training set includes the number of serviceable servers. Our system has two servers denoted as "serverone" and "servertwo". For example, if a user wants to log in to both the servers, then the output will be "serverboth". Once the training process is over, the TA sends *ID_i* to the user.

3.2.2. Login Phase

In the login phase, suppose that *U_i* wants to log in to a server. The steps are as follows:

- The user obtains a time sequence T, which is like a time stamp.
- Then, the user computes W as such

$$W = g^{pw_T} \bmod p$$

- Afterwards, the user delivers *ID_i*, W and T to the server.

3.2.3. Validation Phase

In the validation phase, the server receives W , ID, T at time T^{t}. Now, the server performs the following tasks to validate the user login request:

- First, the server calculates ΔT. ΔT denotes the expected legal time interval for the transmission delay between the login terminal and the system servers. Then, the server checks the validity of the time stamp. If the time interval between T and T^{t} is greater than ΔT, the server rejects the request.
- If the time stamp T is within the valid period ΔT, the server decrypts ID_i using the shared secret key

  $$pw = D_k (ID_i).$$
- The Server after obtaining the password pw, verifies if the following equation holds

  $W = g^{pw_T} \bmod p$.
- If the previous verification holds, the server validates the user. To provide service to the user, the server first maps each of the eight characters of the password into a value according to the mapping table. Then, the server sends these values as an input to the neural network to obtain the output. The output represents that the user is authorized to use the server.

3.3. Experimentation

In the experimentation phase, we used the training pattern to test the learning ability and performance of our neural network model. Our method is a supervised learning method. We used the python programming language to implement the model. We also used the sklearn machine learning library which provides state-of-the-art machine learning algorithms. We assumed the multiserver has two servers denoted as "serverone" and "servertwo". Using our remote password authentication scheme, the user can log in to any one of the servers or both the servers. We also assumed that our multiserver authentication scheme has 540 users. Each user password consists of eight characters. Therefore, our neural network model has eight inputs and one output. The output of the neural network consists of “serverone”, “servertwo” and “serverboth”. For instance, the password of U_1 is “kf12ghty” and the user is granted registration for server one and server two. Then, the expected output of the neural network will be “serverboth”.

3.3.1. Training

Our neural network is based on the multilayer perceptron model (MLP) [48]. Each training pattern has 9 values. The feature vectors are each represented by “a”, “b”, “c”, “d”, “e”, “f”, “g”, “h”. In this experiment, we assume that we have 540 users and two servers. Hence, there are 8 inputs and one output. The training set is then the input to train the network.

3.3.2. Classification

After the training phase, each server validates if the *pw*, *ID* and *W* are correct. If *ID*, *pw* and *W* are correct, the server accepts the request. Then, the server will input the password as feature vectors to compute the classification output. To compare the accuracy and performance of our neural network we used different classifiers. The classifiers that we used are as follows [49]:

- GaussianNB is based on Naive Bayes Methods. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of independence between every pair of features. GaussianNB implements the Gaussian Naive Bayes algorithm for classification.
- Decision Tree is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Support Vector machine is denoted as SVM and is a set of supervised learning methods used for classification, regression and outlier detection. These SVM are effective in high dimensional spaces as different kernel functions can be specified for the decision functions. In addition, they are versatile too. Support Vector Classifiers supports kernels

like “rbf”, “sigmoid”, “poly” and “linear”. Table 3.1 list the different instances used with their corresponding parameters.

- Gradient Boosting (GB) is a machine learning technique for regression and classification. GB builds an additive model in a forward stage-wise fashion.
- A Random Forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the data set and uses averaging to improve the predictive accuracy and controls over-fitting.
- The Extra Tree class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the data set and uses averaging to improve the predictive accuracy and controls over-fitting.
- The Logistic Regression class implements regularized logistic regression using the ‘liblinear’ library, ‘newton-cg’, ‘sag’, and ‘lbfgs’ solvers. It can handle both dense and sparse input.
- K-nearest Neighbors implements learning based on the k-nearest neighbors of each query point, where k is an integer value specified by the user.

3.4. Evaluations and Results

For the experiments, we evaluated the accuracy of the different classifiers mentioned above. Each classifier is executed ten times and then the average of them was taken. Then, we compared the results of each classifier. Table 3.2 shows the first ten iterations of different classifiers.

Table 3.1. SVM instances with corresponding parameters

Classifier	Parameters
SVC 1	kernel="rbf",C=1,=0.8
SVC 2	kernel="rbf",C=0.001,=1
SVC 3	kernel="sigmoid",C=1,=0.8
SVC 4	kernel="sigmoid",C=0.001,=1
SVC 5	kernel="Linear",C=1
SVC 6	kernel="Linear",C=0.001
SVC 7	kernel="poly",C=1,=0.8,degree=3
SVC 8	kernel="poly",C=0.001,=1,degree=3
SVC 9	kernel="poly",C=0.001,=1,degree=2
SVC 10	kernel="poly",C=0.001,=1,degree=1
LinearSVC 1	LinearSVC,multi-class="ovr",C=1
LinearSVC 2	LinearSVC,multi-class="crammer",C=11

Table 3.2. Classification accuracy of different classifiers - Iterations 1-10

Iterations	1	2	3	4	5	6	7	8	9	10
GaussianNB	0.8657	0.8287	0.6829	0.7617	0.8333	0.8218	0.7708	0.7708	0.8426	0.8542
DecisionTree	0.9796	0.8981	0.8611	0.8519	0.8125	0.8241	0.8704	0.8567	0.9120	0.8519
GradientBoosting	0.9097	0.9097	0.8843	0.9097	0.9190	0.8912	0.8912	0.8889	0.9097	0.9097
RandomForest	0.9144	0.9120	0.9213	0.9167	0.8958	0.8912	0.8981	0.9028	0.9144	0.9028
ExtraTree	0.8634	0.8889	0.8958	0.8727	0.8819	0.8773	0.8750	0.9028	0.9020	0.8704
LogisticRegression	0.9051	0.9005	0.9236	0.9097	0.9120	0.9213	0.9329	0.9259	0.9051	0.9282
Kneighbors	0.9120	0.9213	0.8843	0.9074	0.8750	0.8819	0.9167	0.9074	0.8958	0.9120
LinearSVC 1	0.8380	0.5116	0.9282	0.7986	0.8958	0.7593	0.9144	0.8333	0.9190	0.4722
LinearSVC 2	0.6898	0.7824	0.1806	0.6227	0.4074	0.6481	0.5579	0.1389	0.5486	0.3079
SVC 1	0.8935	0.9074	0.8750	0.9074	0.8843	0.8634	0.8981	0.9074	0.8704	0.8889
SVC 2	0.9282	0.0602	0.0255	0.9259	0.0579	0.0440	0.9282	0.0208	0.0463	0.0532
SVC 3	0.9282	0.0162	0.9282	0.0417	0.0208	0.0162	0.0231	0.0255	0.0556	0.0440
SVC 4	0.0622	0.9282	0.0162	0.0208	0.0255	0.0519	0.0532	0.0208	0.0509	0.9236
SVC 5	0.6111	0.7824	0.8264	0.6644	0.7917	0.7014	0.7037	0.6296	0.4907	0.2523
SVC 6	0.3889	0.4468	0.5602	0.6088	0.4792	0.7616	0.4931	0.5556	0.4190	0.4884
SVC 7	0.7593	0.7639	0.8079	0.8218	0.8356	0.8380	0.7963	0.7083	0.8218	0.8148
SVC 8	0.8819	0.8333	0.8241	0.8657	0.8380	0.8843	0.9005	0.7454	0.8912	0.8356
SVC 9	0.8611	0.8472	0.8102	0.8750	0.8218	0.8773	0.8310	0.8981	0.8727	0.8611
SVC 10	0.5949	0.5972	0.6574	0.4722	0.4884	0.4792	0.6505	0.5718	0.5532	0.5463

Since Support Vector Classifiers (SVC) have different kernels, we have listed all the kernels in our classification table (Table 3.3). Table 3.3 shows the average iterations of all the classifiers mentioned above. From Table 3.3, we can see that Gradient Boosting, Random Forest, Logistic Regression and Kneighbors outperforms the other classifiers. Fig. 3.2 shows the box plot of the classifiers accuracy with their standard deviations. The figures gives us a sketch of significance difference of the classifiers. However, to get a detailed understanding of the statistical significance we used the Tukey's significance difference test.

Table 3.3. Classification accuracy of different classifiers

Classifiers	**Accuracy**
GaussianNB	0.80324
Decisian Tree	0.86180
Gradient Boosting	0.90231
Random Forest	0.90695
Extra Tree	0.88310
Logistic Regression	0.91643
KNeighbors	0.90138
LinearSVC 1,c=1	0.78700
LinearSVC 2,c=1	0.48840
SVC 1	0.88950
SVC 2	0.30902
SVC 3	0.20995
SVC 4	0.37466
SVC 5	0.69537
SVC 6	0.52016
SVC 7	0.79677
SVC 8	0.85000
SVC 9	0.85550
SVC 10	0.56110

Table 3.4 shows the way the classifiers are grouped into different classes. From the table, we can say that the means which do not share the same classes are statistically significantly different. Clas- sifiers such as Logistic Regression, Random Forest, Gradient Boosting, Kneighbors, SVC 1, Extra Tree, Decision tree, SVC 10, SVC 8, GaussianNB, SVC 7, LinearSVC 1, SVC 5 are statistically significantly different from SVC 2, SVC 4 and SVC 3. Also GaussianNB, SVC 7, LinearSVC 1 are statistically significantly different from LinearSVC 2, SVC 2, SVC 4 and SVC 3. Table 3.5, 3.6 and 3.7 shows the pairwise comparison of different classifiers with respect to their statistical significance difference and *p* value. We denoted the statistical significant difference as "positive" if their *p* value is less than or equal to 0.005.

3.5. Security Analysis

In this section, we analyzed the security part of our neural network model. Since our new remote authentication method is based on the generation and distribution of the keys, we should be aware that there can be possible attacks on our neural network model. In this section, we will investigate privacy, nonforgability and replay resistance.

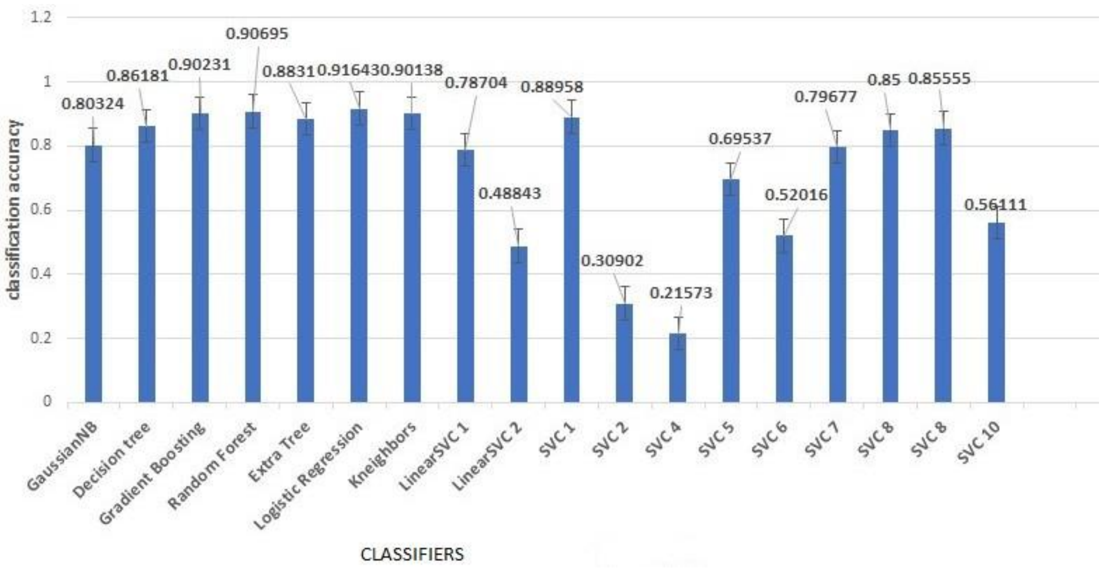

Figure 3.2. Classification accuracy chart of different classifiers

3.5.1. Privacy

In our authentication scheme we used the Diffie Hellman Key (DHK) exchange protocol. To keep the password secret and yet transfer the password to the server and TA, the generation of the keys is done by the DHK exchange protocol. A third party can try to change the shared secret key while the user and the server are communicating (man-in-the-middle-attack). Then, the third party has to change the key while the server and TA or TA and the User are communicating. It is not possible for the same third party to be present in both places at one time. Also, the third party has to continue to be in the middle. If the third party is ever absent, then the user and the server will get to know about his/her presence. Also, for our security the key is established once.

We choose our prime number p to be very large, so that it is not possible for any adversary to traverse through all the prime numbers in polynomial time. The message W cannot be broken. The security benefit comes from the difficulty of solving the discrete logarithm [50] problem. Also, the trusted authority gives the key only to the server/servers the user wants to log in. If the user wants to log in to a different server, the user again has to go through the registration phase to receive a different key.

Table 3.4. Grouping information of different classifiers

Factor	**N**	**Mean**	**Class**
Logistic Regression	10	0.91643	A
Random Forest	10	0.90695	A
Gradient Boosting	10	0.90231	A
Kneighbors	10	0.90138	A
SVC 1	10	0.88958	A
Extra Tree	10	0.88310	A
Decision tree	10	0.86181	A
SVC 10	10	0.85555	A
SVC 8	10	0.85000	A
GaussianNB	10	0.80320	A,B
SVC 7	10	0.79680	A,B,C
LinearSVC 1	10	0.78700	A,B,C
SVC 5	10	0.69540	A,B,C,D
SVC 9	10	0.56110	B,C,D,E
SVC 6	10	0.52020	C,D,E
LinearSVC 2	10	0.48840	D,E,F
SVC 2	10	0.30900	E,F
SVC 4	10	0.21600	F
SVC 3	10	0.21000	F

3.5.2. Nonforgability

A legal user wants to access a non-serviceable server. The server will accept the *ID* to decrypt it. Then, the server will provide it as input to the neural network to receive its output. If the output is different then the server will not accept the request. Hence, our new remote server authentication/validation scheme is nonforgeable.

3.5.3. Replay Resistance

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it. To prevent the replay attack our scheme associates a time T with ID. When an adversary replays a intercepted message to pretend a valid user, he/she has to pass Step 1 of the validation phase. The adversary will create a timestamp $T*$ such that $T" - T* \leq \Delta T$. Once T is changed to $T*$ then W changes and hence the password, thus the adversary will fail the validation phase. However, if ΔT is too large, the server cannot resist the replay attack. Hence, it is very important to choose an appropriate ΔT

Table 3.5. Difference of means to calculate statistical significant difference (STD)

Levels	**Means**	**P value**	**Difference (STD)**
Decision tree - GaussianNB	0.0586	1	Negative
Gradient Boosting - Gaussian NB	0.0991	0.99896	Negative
Random Forest - Gaussian NB	0.1037	0.99815	Negative
Extra Tree - GaussianNB	0.0799	0.99995	Negative
Logistic Reg - GaussianNB	0.1132	0.99471	Negative
Kneighbors - GaussianNB	0.0981	0.99908	Negative
LinearSVC 1 - GaussianNB	-0.0162	1	Negative
LinearSVC 2 - GaussianNB	-0.3148	0.01077	Positive
SVC 1 - GaussianNB	0.0863	0.99984	Negative
SVC 2 - GaussianNB	-0.4942	5.8E-05	Positive
SVC 3 - GaussianNB	-0.5933	5.8E-05	Positive
SVC 4 - GaussianNB	-0.5875	5.8E-05	Positive
SVC 5 - GaussianNB	-0.1079	0.99701	Negative
SVC 6- GaussianNB	-0.2831	0.04199	Positive
SVC 7 - GaussianNB	-0.0065	1	Negative
SVC 8 - GaussianNB	0.0468	1	Negative
SVC 9 - GaussianNB	0.0523	1	Negative
SVC 10 - GaussianNB	-0.2421	0.17933	Negative
Gradient Boosting-Decision tree	0.0405	1	Negative
Random Forest - Decision tree	0.0451	1	Negative
Extra Tree - Decision tree	0.0213	1	Negative
Logistic Reg - Decision tree	0.0546	1	Negative
Kneighbors - Decision tree	0.0396	1	Negative
LinearSVC 1 - Decision tree	-0.0748	0.99998	Negative
LinearSVc 2 - Decision tree	-0.3734	0.00061	Positive
SVC 1 - Decision tree	0.0278	1	Negative
SVC 2 - Decision tree	-0.5528	5.8E-05	Positive
SVC 3 - Decision tree	-0.6519	5.8E-05	Positive
SVC 4 - Decision tree	-0.6461	5.8E-05	Positive
SVC 5 - Decision tree	-0.1664	0.81444	Negative
SVC 6 - Decision tree	-0.3417	0.003	Positive
SVC 7 - Decision tree	-0.0650	1	Negative
SVC 8 - Decision tree	-0.0118	1	Negative
SVC 9- Decision tree	-0.0063	1	Negative
SVC 10 - Decision tree	-0.3007	0.02017	Positive
Random Forest - Gradient Boosting	0.0046	1	Negative
Extra Tree - Gradient Boosting	-0.0192	1	Negative
Logistic Reg - Gradient Boosting	0.0141	1	Negative
Kneighbors - Gradient Boosting	-0.0009	1	Negative
LinearSVC 1 - Gradient Boosting	-0.1153	0.99348	Negative
LinearSVC 2 - Gradient Boosting	-0.4139	0.00011	Positive
SVC 1 - Gradient Boosting	-0.0127	1	Negative
SVC 2 - Gradient Boosting	-0.5933	5.8E-05	Positive
SVC 3 - Gradient Boosting	-0.6924	5.8E-05	Positive
SVC 4 - Gradient Boosting	-0.6866	5.8E-05	Positive
SVC 5 - Gradient Boosting	-0.2069	0.44581	Negative
SVC 6 - Gradient Boosting	-0.3821	0.0004	Positive
SVC 7 - Gradient Boosting	-0.1055	0.99771	Negative
SVC 8 - Gradient Boosting	-0.0523	1	Negative
SVC 9 - Gradient Boosting	-0.0468	1	Negative
SVC 10 - Gradient Boosting	-0.3412	0.00307	Positive
Extra Tree - Random Forest	-0.0239	1	Negative
Logistic Reg - Random Forest	0.0095	1	Negative
Kneighbors - Random Forest	-0.0056	1	Negative

Table 3.5. Difference of means to calculate statistical significant difference (STD) (continued)

Levels	Means	P value	Difference (STD)
LinearSVC 1 - Random Forest	-0.1199	0.98986	Negative
SVC 1 - Extra Tree	-0.5741	0.000058	Positive
SVC 3 - Extra Tree	-0.6732	0.000058	Positive
SVC 4 - Extra Tree	-0.6674	0.000058	Positive
SVC 5 - Extra Tree	-0.1877	0.630259	Negative
SVC 6 - Extra Tree	-0.3629	0.001031	Positive
SVC 7- Extra Tree	-0.0863	0.999837	Negative
SVC 8- Extra Tree	-0.0331	1	Negative
SVC 9 - Extra Tree	-0.0275	1	Negative
SVC 10 - Extra Tree	-0.3220	0.007728	Positive
Kneighbors - Logistic Reg	-0.0151	1	Negative
LinearSVC 1 - Logistic Reg	-0.1294	0.977455	Negative
LinearSVC 2- Logistic Reg	-0.4280	0.00008	Positive
SVC 1 - Logistic Reg	-0.0269	1	Negative
SVC 2 - Logistic Reg	-0.6074	0.000058	Positive
SVC 3 - Logistic Reg	-0.7065	0.000058	Positive
SVC 4 - Logistic Reg	-0.7007	0.000058	Positive
SVC 5 - Logistic Reg	-0.2211	0.32243	Negative
SVC 6 - Logistic Reg	-0.3963	0.000209	Positive
SVC 7 - Logistic Reg	-0.1197	0.990088	Negative
SVC 8 - Logistic Reg	-0.0664	0.999997	Negative
SVC 9 - Logistic Reg	-0.0609	0.999999	Negative
SVC 10 - Logistic Reg	-0.3553	0.001515	Positive
LinearSVC 1 - Kneighbors	-0.1143	0.994052	Negative
LinearSVC 2 - Kneighbors	-0.4129	0.000114	Positive
SVC 1 - Kneighbors	-0.0118	1	Negative
SVC 2 - Kneighbors	-0.5924	0.000058	Positive
SVC 3 - Kneighbors	-0.6914	0.000058	Positive
SVC 4 - Kneighbors	-0.6856	0.000058	Positive
SVC 5 - Kneighbors	-0.2060	0.454487	Negative
SVC 6 - Kneighbors	-0.3812	0.000415	Positive
SVC 7 - Kneighbors	-0.1046	0.997942	Negative
SVC 8 - Kneighbors	-0.0514	1	Negative
SVC 9 - Kneighbors	-0.0458	1	Negative
SVC 10 - Kneighbors	-0.3403	0.003212	Positive
LinearSVC 1- LinearSVC 2	-0.2986	0.022071	Positive
SVC 1- LinearSVC 1	0.1025	0.998396	Negative
SVC 2 - LinearSVC 2	-0.4780	0.000059	Positive
SVC 3 - LinearSVC 1	-0.5771	0.000058	Positive
SVC 4 - LinearSVC 2	-0.5713	0.000058	Positive
SVC 5 - LinearSVC 1	-0.0917	0.999628	Negative
SVC 6 - LinearSVC 2	-0.2669	0.078013	Positive
SVC 7 - LinearSVC 1	0.0097	1	Negative
SVC 8 - LinearSVC 2	0.0630	0.999999	Negative
SVC 9 - LinearSVC 1	0.0685	0.999995	Negative
SVC 10 -LinearSVC 2	-0.2259	0.284575	Negative
SVC 1 - LinearSVC 1	0.4011	0.000171	Positive
SVC 2 - LinearSVC 2	-0.1794	0.707607	Negative
SVC 3 - LinearSVC 1	-0.2785	0.050346	Positive
SVC 4 - LinearSVC 2	-0.2727	0.06285	Negative
SVC 5 - LinearSVC 1	0.2069	0.445806	Negative
SVC 6 - LinearSVC 2	0.0317	1	Negative
SVC 7 - LinearSVC 1	0.3083	0.01442	Positive
SVC 8 - LinearSVC 2	0.3616	0.001105	Positive

Table 3.5. Difference of means to calculate statistical significant difference (STD) (continued)

Levels	**Means**	**P value**	**Difference (STD)**
SVC 9 - LinearSVC 1	0.3671	0.000835	Positive
SVC 10 - LinearSVC 2	0.0727	0.999987	Negative
SVC 2 - SVC 1	-0.5806	0.000058	Positive
SVC 3 - SVC 1	-0.6796	0.000058	Positive
SVC 4 - SVC 1	-0.6738	0.000058	Positive
SVC 5 - SVC 1	-0.1942	0.567748	Negative
SVC 6 - SVC 1	-0.3694	0.000743	Positive
SVC 7 - SVC 1	-0.0928	0.999561	Negative
SVC 8 - SVC 1	-0.0396	1	Negative
SVC 9 - SVC 1	-0.0340	1	Negative
SVC 10 - SVC 1	-0.3285	0.00569	Positive
SVC 3 - SVC 2	-0.0991	0.998964	Negative
SVC 4 - SVC 2	-0.0933	0.999529	Negative
SVC 5 - SVC 2	0.3863	0.000324	Positive
SVC 6 - SVC 2	0.2111	0.407299	Negative
SVC 7 - SVC 2	0.4878	0.000058	Positive
SVC 8 - SVC 2	0.5410	0.000058	Positive
SVC 9 - SVC 2	0.5465	0.000058	Positive
SVC 10 - SVC 2	0.2521	0.130564	Negative
SVC 4 - SVC 3	0.0058	1	Negative
SVC 5 - SVC 3	0.4854	0.000059	Positive
SVC 6 - SVC 3	0.3102	0.013264	Positive
SVC 7 - SVC 3	0.5868	0.000058	Positive
SVC 8 - SVC 3	0.6400	0.000058	Positive
SVC 9 - SVC 3	0.6456	0.000058	Positive
SVC 10 -SVC 3	0.3512	0.001869	Positive
SVC 5 - SVC 4	0.4796	0.000059	Positive
SVC 6 - SVC 4	0.3044	0.017145	Positive
SVC 7 - SVC 4	0.5810	0.000058	Positive
SVC 8 - SVC 4	0.6343	0.000058	Positive
SVC 9 - SVC 4	0.6398	0.000058	Positive
SVC 10 - SVC 4	0.3454	0.002494	Positive
SVC 6 - SVC 5	-0.1752	0.744437	Negative
SVC 7 - SVC 5	0.1014	0.998607	Negative
SVC 8 - SVC 5	0.1546	0.890284	Negative
SVC 9 - SVC 5	0.1602	0.857492	Negative
SVC 10 - SVC 5	-0.1343	0.967608	Negative
SVC 7 - SVC 5	0.2766	0.054134	Positive
SVC 8 -SVC 5	0.3298	0.00533	Positive
SVC 9 -SVC 5	0.3354	0.004078	Positive
SVC 10 - SVC 5	0.0410	1	Negative
SVC 7 - SVC 6	0.0532	1	Negative
SVC 8 - SVC 6	0.0588	1	Negative
SVC 9 - SVC 6	-0.2357	0.217453	Negative
SVC 10 - SVC 6	0.0056	1	Negative
SVC 9 -SVC 8	-0.2889	0.03319	Positive
SVC 10 - SVC 8	-0.2944	0.026349	Positive

3.6. Summary

In this chapter, we designed a remote password authentication scheme for multiserver environments. The user will register first with a trusted authority to become a legitimate user. Each legitimate user has its own ID and password. Then, the user types his/her user ID and password to log in to any of the servers. Later, during the authentication phase the servers validate the legitimacy of the remote login user. The remote password authentication scheme for multiserver environments is based on artificial neural networks. To overcome the problem of the man-in-the- middle-attack we created our shared secret key using the three party Diffie Hellman Key exchange protocol.

For the experimentation, we use 540 users with different passwords of eight characters. Each of these eight characters represent the input feature vectors of our neural network. The output represents a number of serviceable servers. Each user can log in to any one of the servers or both the server at one time. After training the neural network, we compared different classifiers in terms of efficiency and accuracy.

Our findings shows that Gradient Boosting, Random Forest, Logistic Regression and K neighbors outperforms the other classifiers. However, to identify whether a classifier is statistically significantly different from the other we used the Tukeys test method. The test method found that Logistic Regression, Random Forest, Gradient Boosting and Kneighbors are not statistically significantly different from SVC 1, Extra Tree, Decision tree, SVC 8, SVC 10 and Gaussian NB. From Tables 3.5, 3.6 and 3.7 we get a more accurate picture of statistical significant difference of each classifier as compared to other classifier.

4. IMPROVING DATA PRIVACY USING FUZZY LOGIC AND AUTOENCODER NEURAL NETWORK

Data privacy is a very important problem while sharing data among multiple organizations and has become a very crucial problem in the health sectors since multiple organizations such as hospitals are storing data of the patients in the form of Electronic Health Records. Stored data is used with other organizations or research analysts to improve the health care of patients. However, the data records contain sensitive information such as age, sex, and date of birth of the patients. Revealing sensitive data can cause a privacy breach of the individuals. This has triggered research that has led to many different privacy preserving techniques being introduced. Thus, we designed a technique that not only encrypts / hides the sensitive information but also sends the data to different organizations securely. To encrypt sensitive data we use different fuzzy logic membership functions. We then use an autoencoder neural network to send the modified data. The output data of the autoencoder can then be used by different organizations for research analysis. The advantage of using an autoencoder neural network is that it learns the input feature vector.

4.1. Related Work

There has been a surge in recent research activity in privacy preserving of sensitive data. Several papers have been published on various aspects of privacy preserving. We discuss here some of the previous work related to our approach.

Several data hiding technique exists on different assumptions. Data swapping is the commonly used data hiding technique. It refers to a method of swapping information from one record to another [51],[52]. The amount of swapping to be done in a database is dependent on the ap- plication and the need of the organization. There exist various variants of swapping. The records for swapping are purposely or randomly chosen since they are believed to have a greater risk of re-identification. The advantage of swapping is that it can be easily implemented and is one of the best methods of preserving confidentiality. Its main disadvantage is that even with a very low swapping rate, it can destroy analytic properties, particularly sub-domains.

To overcome the disadvantage of the earlier two methods, the authors introduced a controlled way of swapping in 1995 [53]. In this approach, the values of an individual record are sorted and swapped in a range of k-percent of the total range. Randomization determines the specific values of the record value to be swapped. The procedure is repeated for each variable until all variables have been swapped. The main disadvantage of this approach is the determination of k. If k is relatively small, then analytic distortions on the entire file may be small for simple regression analysis. If k is large, there is an assumption that the re-identification risk may be reduced [54]. The authors provided methods for aggregating several attributes simultaneously. The methods are based on multi-variable metrics for clustering variables into the most similar groups. The methods are not as easily implemented because they can involve sophisticated optimization algorithms. For computational efficiency, the methods are applied from two to four attributes simultaneously whereas many public use files contain 12 or more attributes. The advantage of the multi-variable aggregation method is that it provides better protection against re-identification. Its disadvantage is that analytic properties can be severely compromised, particularly if two or three uncorrelated attributes are used in the aggregation process. The attributes that are not micro-aggregated may themselves allow re-identification.

Micro-aggregation is another technique for data masking [56],[55]. It aggregates the record values of attributes and is intended to reduce the re-identification risk. In single ranking micro- aggregation, each attribute is aggregated independently of other attributes. The method is easy to implement and the values of attributes are sorted and divided into groups of size k. In practice, k is taken to be three or four to reduce analytic distortions. In each group, the values are replaced by an aggregate such as the mean or the median. The micro-aggregation is repeated for each of the attributes that are considered to be usable for re-identification.

Adding noise to the data sets showed that it is theoretically possible to recover the mean and covariance of a given record for arbitrary sub-domains [91],[58]. Both of the papers showed that the masked data set by adding noise provides good analytic properties such as regression analysis that closely reproduce regression analysis of the unmasked data. The authors reasoned that adding noise can yield files with moderate re-identification rates. In 2017, the authors [87] used symmetric key encryption to hide the data before sending it to the other party using neural network. In 2018, the same authors [59] used the hiding technique to encrypt the password to log in to multiple servers. However, both encryption approaches used the symmetric key method. In symmetric key encryption the shared secret key has to be secured. In 2017, the authors in [60] came up with the idea of adding noise to the data set using neural networks. The authors achieved privacy by hiding two of the attributes. The disadvantages of using this approach is that they classified age into groups, i.e., two different ages will fall into the same group.

In [61], the authors introduced fuzzy logic for privacy preserving. They claim their technique is useful for both numerical and categorical attributes. However, the authors did not use any data mining technique to prove that their modified data is the same as the raw data. The authors in [62] published a paper showing a comparative study of data perturbation They also used fuzzy logic to preserve privacy. The authors used different classifiers like SVM, ID3 and C4.5 on the original as well as on the perturbed data.

Though different techniques have been applied to preserve privacy, however, each of them has their own limitations. First of all, the data swapping method has a very low swapping rate. Then, in order to overcome the randomization problem, micro-aggregation was used. Both of these techniques have a limitation in determining the split of the data records. In order to avoid this limitation, a data set was created by adding noise using fuzzy logic and neural networks. The authors of the papers did not show how to send the data securely to different other organizations. In addition, the authors added noise only to the attributes that are numerical and categorical.

In our dissertation, we used fuzzy membership functions and an Autoencoder neural network to modify the sensitive information. Our new technique not only hides numerical and categorical attributes but also real attributes. After the fuzzification of three sensitive data, we send the modified data to an Autoencoder neural network. The current approach retains both privacy and the accuracy of the result. Our dissertation is different from the earlier work done so far by combining fuzzy logic with an autoencoder neural network.

4.2. Background

This section explains the background information used in our proposed approach. In our proposed approach we used fuzzy membership functions [63] and an Autoencoder Neural Network.

4.2.1. Fuzzy Logic

Fuzzy logic was introduced in 1965 by Zadeh in his research paper "Fuzzy Sets" [64]. He is considered as the father of fuzzy logic. Fuzzy logic resembles the human decision-making methodology by dealing with vague and imprecise information. Fuzzy logic is applied to many real-world problems since it is based on degrees of truth rather than based on Boolean logic. Fuzzy logic is best understood within the context of set membership. Basically fuzzy logic allows partial membership, which means that it contains elements that have varying degrees of membership within the set. Furthermore, membership functions characterize fuzziness whether the elements in the fuzzy sets are discrete or continuous. We have used different membership functions in our approach, which are explained below.

4.2.2. Mathematical Notation

Here, we elaborate of the different membership functions which have been used to perturb the raw data. Figure 5.1 shows the plot of different membership functions.

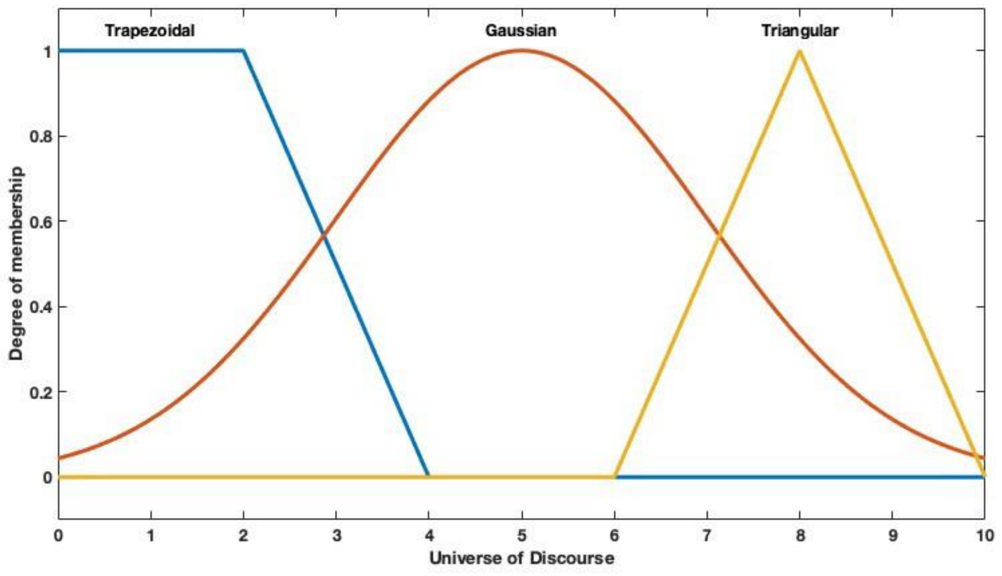

Figure 4.1. Different membership functions

- Triangle Membership Function: This membership function uses the following relationship

$$\text{TriMF}(D : X, Y, Z) = 0 \text{ when } D < X$$

$$= (D - X)/(Y - D) \text{ when } X \leq D \leq Y$$

$$= (Z - D)/(Z - Y) \text{ when } Y \leq D \leq Z$$

$$= 0 \text{ when } D > Z$$

Here, *D* represents the value in the data set. *X*, *Y* and *Z* are three boundary points.

- Gaussian Membership Function: This membership function has the following relation

$$\text{GaussMF}(D : C, W) = \text{EXP} -(D - C)^2/W^2$$

In the above relation, *D* is the value in the data set. *C* is the center, and *W* is the width of the function.

- S-Shaped Membership Function: For this membership function we have the following relation

$$SMF(D : X, Y) = 0 \text{ when } D \leq X$$

$$= 2 * [(D - X)/(Y - X)]^2 \text{ when } X \leq D \leq (X + Y)/2$$

$$= 1 - 2 * [(D - Y)/(Y - X)]^2 \text{ when } (X + Y)/2 \leq D \leq Y$$

$$= 1 \text{ when } D \geq Y$$

where D is the value in the data set. X is the minimum, and Y is the maximum value in the data set.

Our model is based on privacy and security. Hence we kept the boundary points in triangular membership functions, center and the width parameters in Gaussian Membership function and maximum and minimum values in s-shaped membership function secured. These values are know only to the sender and the receiver.

4.2.3. Artificial Neural Network

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of the name suggests, the networks are brain-inspired systems, which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. Artificial neural networks are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract or to teach the machine to recognize. While neural networks (also called "Perceptrons") have been around since the 1940s, it is only in the last several decades that they have become a major part of artificial intelligence.

4.2.4. Autoencoder

In our work we used an autoencoder [65] neural network. The Autoencoder has a multilayer percepepton (MLP) like structure with input layer, hidden layer and output layer. However, unlike an MLP, Autoencoder neural networks do not require any target data since the network tries to learn the input itself. The Autoencoder consists of two parts: encoder and decoder. The encoder compresses the inputs to the most important features. The decoder reconstructs the original input from the encoder. The hidden layer compresses the input of the most important feature vectors. Therefore, the number of feature vectors are reduced in the hidden layer. Autoencoders are very frequently used for dimensionality reduction and feature selections.

Our proposed model is divided into two tasks. The first task is based on hiding the sensitive information. The second task is to send the perturbed data to different organization using an Autoencoder. Thus, in our proposed approach in order to accomplished the first task we used fuzzy membership functions to hide the sensitive attributes. These are the attributes which the patient does not want to share. By hiding the data using different fuzzy membership functions this will make it difficult for some other party to identify the patient. Next, to send data to different organizations we used an Autoencoder. By using an Autoencoder we can keep the raw data and the perturbed data set almost similar otherwise no other organization will be interested in a totally perturbed data set.

4.3. Our Approach

In our approach, we focused on how to improve the data privacy of the patients. Data pri- vacy could be improved if we could hide the sensitive information of the patient before sending the data to different organizations. Also, the modified data and the raw data should almost be same. Otherwise the modified data set will be misleading. The research analyst will not be interested in using such a modified data set.

In our approach, we selected a recent cervical cancer (risk factor) data set from UCI data repository [66]. This data set focuses on the prediction of indicators / diagnosis of cervical cancer. The data set was collected by the “Hospital Universitario de Caracas” in Caracas, Venezuela. The data set compromises demographic information, habits and historic medical records of 858 patients. The 36 attributes are boolean or real valued. The data set has 4 target variables, which are Hinselmann, Schiller, Cytology, Biopsy.

We first formatted the data set. As an Autoencoder learns the feature vector and reconstructs the output, we reformatted the data set. We included the target variables in the feature. We then sent all the features as the feature vector to an Autoencoder. We modified the data set using the following ways:

- First we identified the sensitive attributes from the data set. We identified age, number of sexual partners, and first sexual intercourse as sensitive information. This is the information which most of the patients do not want to share since their identity could be revealed by sharing this information.
- We perturbed or hide these three attributes using three fuzzy membership functions. The three fuzzy membership functions are S-shaped, Gaussian and Triangular. These three membership functions are described in the preliminaries section. These membership functions require boundary points, which are selected by the organization which owns the data set. In our approach, the hospital which has the raw data will select the boundary points. These boundary points should be securely stored by the hospital. The boundary points should not be shared with any other organization since these could reveal the patients' identity. The rest of the attributes are left as they are in the raw data set.

To share the data with different organizations we used an autoencoder neural network. We are not interested in the class which predicts the cervical cancer rather we are interested in the feature vectors as the output. So we consider all the features including the target variables as feature vectors. An auto encoder is one of the neural networks which learns its own input. Thus, we used an autoencoder to send the data set to different other organizations. Fig. 5.2 outlines the autoencoder used in our approach.

From the figure we can see that we used 36 feature vectors as input. The hidden layer compresses the 36 feature vectors to 20. These 20 feature vectors are the most important attributes. The decoder reconstructs those 36 attributes from the encoder. Since the auto encoder neural network is reconstructing the 36 feature vectors again, we measure the accuracy with different loss functions. The output of the auto encoder is sent as a data set to different organizations. The data set shares the same information as the raw data set without compromising the privacy of the patients.

4.4. Experimental Setup

The cervical cancer data set which we used in our experiment consists of 36 attributes. Among them we consider the first three attributes as sensitive. We have hidden those attributes with three fuzzy membership functions. The three fuzzy membership functions which we used are Gaussian, S-shaped and Triangular. Thus, in total we have 42 attributes instead of 36 attributes in our data set.

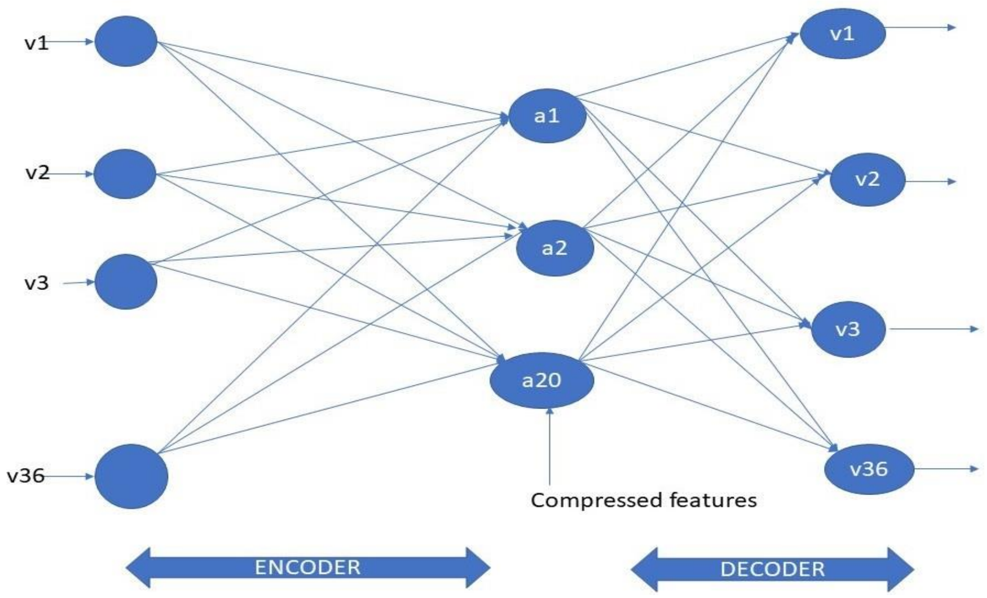

Figure 4.2. Autoencoder neural network

Next, we have to send the modified data set to another party using an Autoencoder. We dropped two columns ‘Time since first diagnosis’ and ‘Time since last diagnosis’ since these two columns have the most missing values. Thus, we are left with 40 columns. Though we have 40 attributes, but at one time we are sending only 34 attributes through the Autoencoder. These 34 attributes exclude the attributes created by different other membership functions. These 34 attributes are the feature vectors/ input to the Autoencoder. The hidden layer/layers tries to compress the input features to a latent space representation. The output is reconstructed from this representation. This is how the output of our Autoencoder is created. The reconstructed output includes all the feature vectors. To obtain a better accuracy of our modified data set we processed the modified data set. The processing has been done in following three ways:

- Dropping the columns which has missing values.
- Dropping the rows which have missing values.
- Replacing the missing values with the mean of that particular column.

4.5. Evaluation and Results

In this section, we summarize our observations and results. We measure the accuracy against different loss functions and also with and without sparsity constraints enabled in the autoencoder. The sparsity constraints regularize the autoencoder [67].

The parameters which are considered are as follows:

- Random State = 150
- Epoch = 500
- Optimizer = Adadelta
- Activation function = tanh
- train_split= 0.8
- activity_regularizers= 0.000001

Table 5.1 shows the accuracy of different membership functions. We considered the modified data set without sparsity constraints by dropping the missing columns. From the table we can see that Logcosh used as the loss function achieves the best value of 81.60% for the three different membership functions. Also, we can say from the values in the table that MSE (Mean Square Error) as a loss function obtains the best accuracy of 81.15% for the S-shaped function.

In Table 5.2, we evaluated the same data set with the sparsity constraints. We can see that the Gaussian membership function with MSE as the loss function obtains the best accuracy of 87.34%.

Then, we evaluated the data set by dropping the rows which have missing values and ran the autoencoder without the sparsity constraints. The results are given in Table 5.3. From the table we can summarize that the Gaussian and Triangular membership functions with hinge value both obtains the best accuracy of 62.51%. Table 5.4 summarizes the evaluation of the data set running the autoencoder with the sparsity constraints. We can see that the Gaussian membership function with logcosh obtains the best accuracy of 62.51%.

Table 4.1. Accuracy and loss value of drop column data set and without sparsity constraints

Membership function	**Loss function**	**Loss value**	**Accuracy**
Gaussian	Mean_Absolute	0.1313	0.7588
Gaussian	MSE	0.1278	0.8020
Gaussian	Categorical_crossentropy	1.0206	0.0207
Gaussian	Logcosh	0.1364	0.8160
Gaussian	Hinge	0.7107	0.2447
S-shaped	Mean_ Absolute	0.1316	0.7835
S-shaped	MSE	0.1272	0.8115
S-shaped	Categorical_crossentropy	1.0756	0.0166
S-shaped	Logcosh	0.1280	0.8127
S-shaped	Hinge	0.7038	0.2606
Traingular	Mean_ Absolute	0.1356	0.7769
Triangular	MSE	0.1273	0.8063
Triangular	Categorical_crossentropy	0.9532	0.0433
Triangular	Logcosh	0.1280	0.8146
Triangular	Hinge	0.6738	0.2142

Table 4.2. Accuracy and loss value of drop column data set and with sparsity constraints

Membership function	**Loss function**	**Loss value**	**Accuracy**
Gaussian	Mean_Absolute	0.1313	0.7409
Gaussian	MSE	0.1260	0.8734
Gaussian	Categorical_crossentropy	0.9451	0.0058
Gaussian	Logcosh	0.1263	0.8122
Gaussian	Hinge	0.7091	0.1135
S-shaped	Mean_Absolute	0.1326	0.8180
S-shaped	MSE	0.1261	0.8457
S-shaped	Categorical_crossentropy	1.0220	0.0291
S-shaped	Logcosh	0.1266	0.8239
S-shaped	Hinge	0.6954	0.1397
Triangular	Mean_Absolute	0.1313	0.8122
Triangular	MSE	0.1278	0.8457
Triangular	Categorical_crossentropy	0.2936	0.0015
Triangular	Logcosh	0.1267	0.8384
Triangular	Hinge	0.7059	0.1266

Table 4.3. Accuracy and loss value of drop row data set with sparsity constraints

Membership function	**Loss function**	**Loss value**	**Accuracy**
Gaussian	Mean_Absolute	4.4915	0.1458
Gaussian	MSE	6.2613	0.3125
Gaussian	Categorical_crossentropy	4.0312	0.3750
Gaussian	Logcosh	4.2236	0.2708
Gaussian	Hinge	1.1200	0.6251
S-shaped	Mean_Absolute	4.3112	0.4565
Sshaped	MSE	4.2674	0.2917
S-shaped	Categorical_crossentropy	5.8769	0.0333
S-shaped	Logcosh	4.4675	0.1667
S-shaped	Hinge	5.5678	0.0000
Triangular	Mean_Absolute	4.2978	0.0833
Triangular	MSE	4.2634	0.2708
Triangular	Categorical_crossentropy	5.7617	0.0208
Triangular	Logcosh	5.5876	0.1875
Triangular	Hinge	1.7665	0.6250

Table 4.4. Accuracy and loss value of drop row data set and without sparsity constraints

Membership function	**Loss function**	**Loss value**	**Accuracy**
Gaussian	Mean_Absolute	4.3342	0.0208
Gaussian	MSE	4.2642	0.2708
Gaussian	Categorical_crossentropy	5.8342	0.0627
Gaussian	Logcosh	4.3421	0.3542
Gaussian	Hinge	4.7213	0.0000
S-shaped	Mean_Absolute	4.3124	0.2292
S-shaped	MSE	4.2343	0.1875
S-shaped	Categorical_crossentropy	6.2212	0.0625
S-shaped	Logcosh	4.3427	0.2083
S-shaped	Hinge	4.7453	0.0417
Triangular	Mean_Absolute	4.2634	0.1458
Triangular	MSE	4.5674	0.1875
Triangular	Categorical_crossentropy	6.0354	0.0417
Triangular	Logcosh	2.2354	0.3750
Triangular	Hinge	4.7123	0.0000

We also evaluated the data set in which the missing values are replaced by the mean of that particular column. We evaluated that data set running the autoencoder without the sparsity constraints. The results are given in Table 5.5. We can summarize that the Gaussian membership function with logcosh as the loss function returns the best value of 62.52%. From Table 5.6 we can say that the Gaussian membership function with MSE and using the sparsity constraints returns the best value of 64.57%.

Table 4.5. Accuracy and loss value of mean data set and without sparsity constraints

Membership function	Loss function	Loss value	Accuracy
Gaussian	Mean_Absolute	0.2648	0.6250
Gaussian	MSE	4.3532	0.2500
Gaussian	Categorical_crossentropy	2.5200	0.0000
Gaussian	Logcosh	1.3600	0.6252
Gaussian	Hinge	2.1500	0.0000
Sshaped	Mean_Absolute	1.3800	0.2086
Sshaped	MSE	1.3600	0.5331
S-shaped	Categorical_crossentropy	2.2000	0.0000
S-shaped	Logcosh	1.3600	0.5993
S-shaped	Hinge	2.1400	0.0000
Traingular	Mean_Absolute	1.3600	0.5397
Triangular	MSE	1.3600	0.5199
Triangular	Categorical_crossentropy	2.2200	0.1192
Triangular	Logcosh	1.3600	0.3543
Triangular	Hinge	2.0900	0.0000

4.6. Summary

This chapter described a technique that encrypts and hides sensitive information but also sends the data to different organizations securely. In order to encrypt sensitive data, our approach used three different fuzzy logic membership functions. We kept the boundary points secure. The boundary points are know only to sender and receiver Then, we used an Autoencoder to learn the input feature vectors of the modified data set which then allows us to send the output of the Autoencoder to share data with other organizations. As for the experiments, we evaluated three types of data sets. The data sets were modified by dropping columns, dropping rows and mean. We then evaluated the accuracy against different loss functions and measured the accuracy of the Autoencoder with and without sparsity constraints. From all the results we evaluated, we found that the 'dropping column data set' and running the Autoencoder with sparsity constraints obtained the best accuracy. From all the results, we can say that the best accuracy we obtain is by using the Autoencoder with sparsity constraints. Among all the fuzzy set functions, the Gaussian membership function with MSE as the loss function using the data set with the dropped column obtained both a very good accuracy and also a low loss value. Hence, the Gaussian membership function can be used to hide / encrypt sensitive information. Also, to send the data we used an Autoencoder with MSE as the loss function and obtained an accuracy of 87.34%.

Table 4.6. Accuracy and loss value of mean data set with sparsity constraints

Membership function	**Loss function**	**Loss value**	**Accuracy**
Gaussian	Mean_Absolute	1.3777	0.1556
Gaussian	MSE	1.0600	0.6457
Gaussian	Categorical_crossentropy	2.9306	0.0000
Gaussian	Logcosh	1.3600	0.2815
Gaussian	Hinge	2.1100	0.0000
S-shaped	Mean_Absolute	1.1382	0.1854
S-shaped	MSE	1.3600	0.4437
S-shaped	Categorical_crossentropy	2.6700	0.0000
S-shaped	Logcosh	1.3600	0.2119
S-shaped	Hinge	2.1300	0.0000
Traingular	Mean_Absolute	1.3770	0.1589
Triangular	MSE	1.3685	0.6159
Triangular	Categorical_crossentropy	2.5670	0.5960
Triangular	Logcosh	1.3693	0.2848
Triangular	Hinge	2.1100	0.0000

In summary, as our technique is based on data privacy we kept the boundary conditions of the Gaussian Membership functions protected. Also for better privacy results the boundary conditions should be used only once. If the hospital has to send data again, they should use different boundary conditions. Thus, the adversary would not be able to guess the correct boundary conditions. Our method can assure patients that their sensitive information will be kept secret, and furthermore, other organizations or data analysts will receive the data that is very similar to the raw data.

5. ANALYZING PRIVACY OF TIME SERIES DATA USING SUBSTITUTE AUTOENCODER NEURAL NETWORK

Nowadays people are very concerned about their health. For example, people want to keep track of how often they are active throughout the day. Hence many smart devices are equipped with sensors which help to keep track of the activities. Examples are smart phones and smart watches, which keep track of jogging, sleeping, sitting, walking, and drinking. Users who use these sensors want to share the information to be analyzed for their own health benefits. However, at the same time the users do not want to share the sensitive information. For example, users might want to share jogging and walking data, but not sleeping and drinking data. In this paper, we propose a privacy preserving predictive model. We used the concept of denoising autoencoder to hide the sensitive attributes of a user. We used two data sets. One is the Skoda data set and the other one is the hand gesture data set. We divided the data sets into three different subsets: desired, sensitive, and non sensitive subsets. The output of the denoising autoencoder will only be the desired and non-sensitive subsets. The sensitive subsets are hidden by the non-sensitive subsets. We evaluated the efficacy of our predictive model using three different flavors of autoencoders. We used CNN autoencoder, Deep autoencoder and LSTM autoencoder. All three autoencoders have the property of denoising autoencoder. To retain the accuracy as similar as possible compared to the original data set, we used a convolutional neural network for classification. We measured the F1-score of our model against each of the three autoencoders. As our predictive model is based on privacy, we have also used a Generative Adversarial Neural Network (GAN), which is used to show to what extend our model is secure.

5.1. Related Work

In this section we describe the various approaches related to our privacy preserving model. The various approaches, which are described here are the differential privacy method [72], the filtering method, data mapping method, and finally the substitution method.

Differential Privacy (DP) [73] is an approach, which adds constraints on the algorithm used to publish aggregate information. DP limits the disclosure of private information of records in the data sets. Furthermore, DP permits companies to access a large number of sensitive data for research analysis and business analysis without privacy breach. Also, research institutions use differential privacy technology to automate privacy processes within cloud-sharing communities across countries. For example, Apple uses DP to protect the privacy of users and resolve data sharing problem. The authors in [74] have shown the amount of noise to be added in differential privacy to make data secure. The authors have shown that by adding gaussian noise instead of laplacian noise will increase the computational efficiency. Also, the authors in [75] used differential privacy with machine learning. The authors used two standard public data sets, MNIST [76] and CIFAR-10 [77]. They improved the privacy of the data sets by introducing the Stochastic Gradient Descent (SGD) [78] algorithm. The major disadvantage of differential privacy is to add more noise if more data has to be hidden. This decreases the efficiency of a data set.

Another approach to maintain privacy in the data is by filtering. Instead of adding constraints on the algorithm, the filtering technique filters the sensitive information. Collaborative filtering is a very secure way to filter the sensitive information. This type of filtering has been best used for giving recommendation to the users. In [79], the authors used collaborative filtering for privacy preserving. They aimed at solving this problem by the systematic collection of sensitive information of preferences. They partitioned the data between parties to ensure privacy. Different techniques of collaborative filtering are mentioned in [80]. The paper gives an overview of the model-based, memory-based and hybrid-based collaborative filtering techniques. The key disadvantage of collaborative filtering is that it does not work well with sparse data sets [81].

Data Mapping is another technique to hide data. Data Mapping is a process to map data fields from the source fields to the target fields. One of the data mapping techniques used in healthcare privacy is geomasking [82]. The authors provide protection for individual addresses while maintaining spatial resolution for mapping purposes using geomasking. In [83], a new adaptive geomasking technique known as donut geomasking was proposed. The new technique extends the current method of geomasking by ensuring a user defined minimum level of geoprivacy. The authors in [84] propose another geographic masking technique known as location swapping. When locations of individual-level health data are released in the form of published maps, the identity of these individuals could be identified through reverse geocoding. Location swapping replaces an original location with a masked location selected from all possible locations with similar geographic characteristics within a specified neighborhood. The main disadvantage of this method that it is impossible to say that the feature extracted from the raw data set does not contain any sensitive information. Another way of perturbing data is known as randomization of data. Randomization of data is a process of making data random. This could be done by generating a random permutation of a sequence, generating a random numbers, or by selecting random samples of the population. The authors in [85] proposed an approach whereby many clients can protect their personal information in a server. The clients can use a randomization algorithm to randomize the data and then sent it to the server. The authors have chosen the randomization technique so that the aggregate properties can be recovered with sufficient precision. The authors in [86] introduce a family of geometric data transformation methods (GDTMs) that distort confidential numerical attributes in order to meet privacy protection in clustering analysis. In [59], the authors used symmetric key encryption to randomize the password. Different classifiers were used to analyze the accuracy of the model. Also, the authors in [87] randomized the sensitive attributes of a patient data set using fuzzy membership [61] functions to hide the sensitive data. However, these randomization methods have not been applied to time series data. The data sets used in all the above mentioned related work are not time series data sets. The authors in [88] introduce the concept of replacement technique with time series data sets. Data replacement or data substitution is a technique that can be used with time series data. The technique replaces the sensitive data of a user by non-sensitive data. Then, the transformed data is sent to the cloud. The authors first divided the data sets into three subsets: desired, sensitive, and nonsensitive. Then, the authors replaced the sensitive attributes with the non-sensitive attributes. The authors applied the denoising autoencoder to transformed the data set. However, the authors limited their approach to only multilayer autoencoder. Thus, in our paper we are experimenting different time series data sets with several different autoencoder neural networks. We showed and compare the efficiency of different flavors of autoencoder in our paper.

5.2. Proposed Approach

This section shows the predictive model to substitute the sensitive attributes of a data set with nonsensitive attributes. The modified data set can be used by a cloud server. Our proposed approach consists of three parties: user, trusted authority, and server. Figure 5.1 shows the high-level architecture of our privacy preserving model.

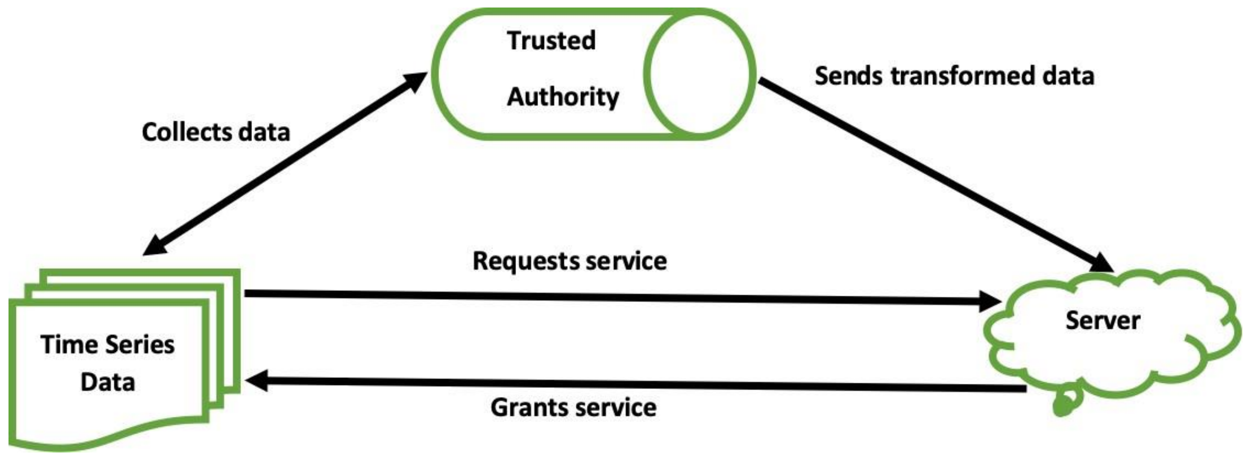

Figure 5.1. A high level architecture: a privacy preserving trusted authority between user and the server

To accomplish the privacy policy of the user we divided the data set. Our data set has been grouped into three subsets: black listed, white listed, and gray listed. The black listed subset consists of sensitive attributes, which the user does not want to reveal. The white listed subset consists of the desired information about the user. This information can benefit the user when shared. The gray listed subset consists of non-sensitive information. The user does not worry about the information if it is shared. Tasks of each three parties are explained below:

User: The user / subject wore a sensor both in the left hand and the right hand. The sensors generate time series data. The data consists of classes ‘write on notepad’, ‘open hood’, ‘close hood’, ‘check gap on the front door’, ‘open left front door’, ‘close left front door’, ‘check trunk gaps’, ‘open and close trunk’, and ‘check steering wheel’. The user wants to receive a benefit from the cloud by sharing this information. However, at the same time the user does not want to share the sensitive attributes.

Trusted Authority: The user trusts only the trusted authority. The trusted authority is a machine learning platform. The user sends the original time series data set to the Trusted Authority. The detailed diagram is given in Figure 5.2. Along with the data set, the user also sends the list of the three subsets. The three subsets consist of the desired, sensitive, and non-sensitive information provided by the user. Inspired by the denoising autoencoder, we implemented the model in our proposed approach. First, we substitute the black listed subset with the gray listed subset. To add more privacy we generated a noisy gaussian digit and clip the images between 0 and 1. Then, we added it to the transformed black listed subset. The transformed training set consists now of the original white listed subset, the original grey listed subset, and the transformed black listed subset. As we substitute the sensitive information with non-sensitive information, we call our autoencoder as substitute autoencoder. Then, we train the substitute autoencoder to map the transformed data set to the original data set.

Server: The server is a third party who needs the desired information for further analysis. The user requests service from the server. The server allows the user to upload the time series data after sharing some information. The user does not trust the server, thus the user does not give the original data set to the server. The user gives the original data set to the trusted authority. The trusted authority is trusted by both the server and the user. The trusted authority transforms the data set by substituting the black subset with the grey subset. The transformed data set is then given to the server.

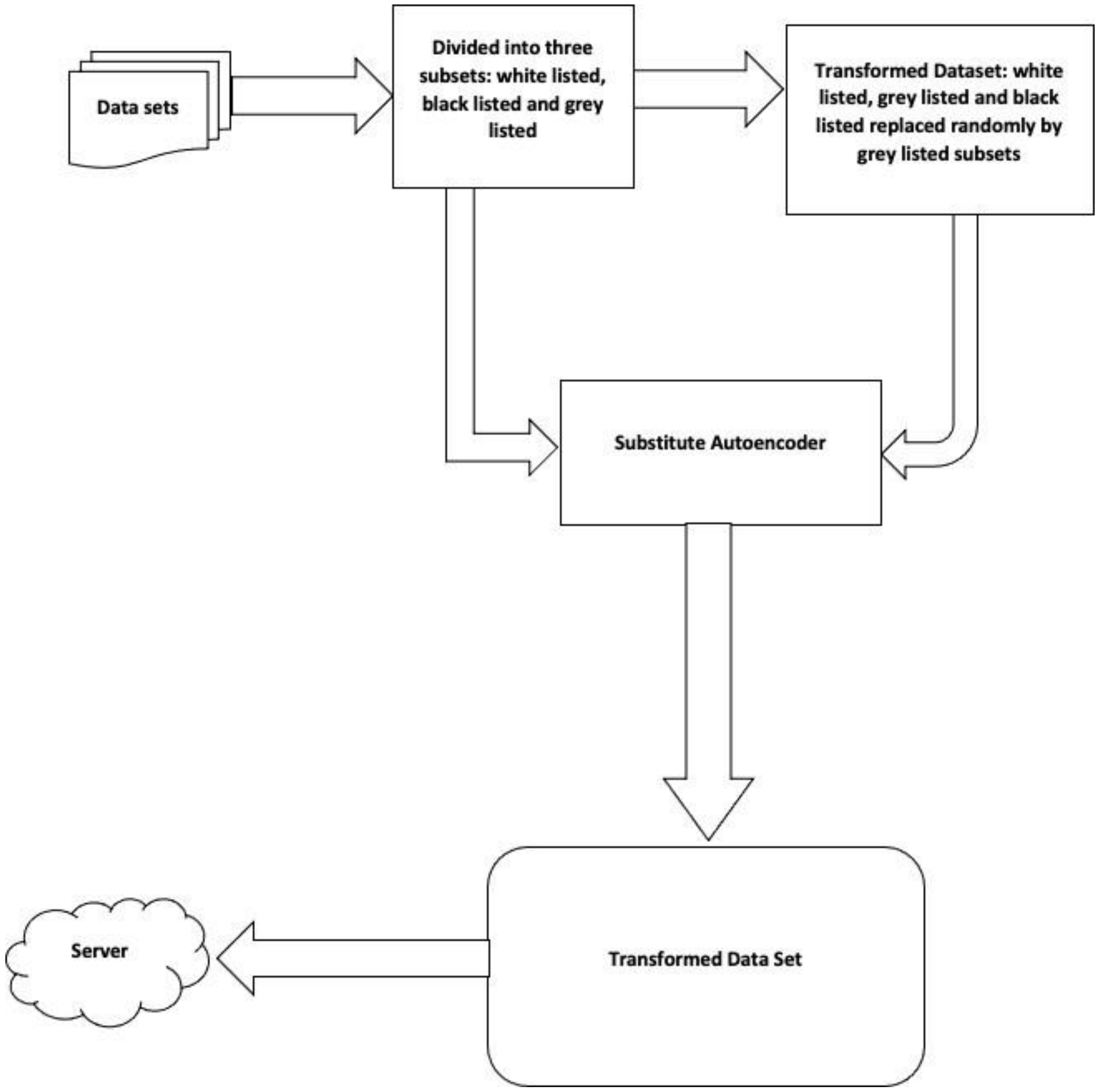
Data sets
Divided into three subsets: white listed, black listed and grey listed
Transformed Dataset: white listed, grey listed and black listed replaced randomly by grey listed subsets
Substitute Autoencoder
Transformed Data Set
Server

Figure 5.2. An overview of the denoising autoencoder process

5.3. Experimental Setup

5.3.1. Data Set

Experiments are conducted on two data sets. The data sets are the Skoda [93] and the Hand gesture data set [94].

- Skoda Data set: This data set describes the activities of assembly-line workers in a car production environment. The data set considers the recognition of 11 activity classes performed for one of the quality assurance checkpoints of the production plan. In the study, one subject wore nineteen 3D accelerometers on both arms and performed a set of experiments using sensors placed on the two arms of a tester (10 sensors on the right arm, and 9 sensor on the left arm). The Skoda data set has been employed to evaluate deep learning techniques in sensor networks, which makes it an appropriate data set to evaluate our proposed substitute autoencoder framework.
- Hand Gesture Data set: In this data set, the sensory data from hand gestures are recorded for two subjects. The data is recorded using an accelerometer and gyroscope worn by the subjects. The data sets consist of 12 classes/activities performed. The 12 classes include 8 regular gestures and 3 gestures for playing tennis. The data set also includes a null activity where no gesture is performed.

5.3.2. Experimental Settings

The Skoda and Hand gesture data sets are human activity and context recognition data sets. The data sets have different classes. We divided the classes into three different subsets: desired, sensitive, and non-sensitive subsets. We mentioned the desired and non-sensitive classes to the server. The user did not provide the sensitive subset to the server but only to the trusted authority. To send the transformed data set to the server, the following steps were performed:

- First we took the sliding window size d = 30 and step size w = 3. Then, each of the classes/activities are mapped to numbers starting from zero. Afterwards, using the train split we created the training and testing data set. The training and testing data sets are in the shape of (samples, features, window size).
- In the second step, we created three data sets: White (desired), Black (sensitive), Gray (non sensitive) subsets. These subsets will be used to train the substitute autoencoder.

- We created a transformed data set, which will hide the sensitive attributes. To hide the sensitive attributes we substitute the non-sensitive or gray data set with the black data set. After substitution, we added gaussian noise to the transformed black subset using the following commands:

```
rnd_idx train = np.random.choice(g train data.shape[0], b train data.shape[0], replace=False)

b_ train_ transformed = g train data[rnd_idx train,:]

b_ train_transformed = x_train_transformed + 0.5 * np.random.normal(loc=0.0, scale=1.0,
size=b_train_transformed.shape)

b_train_transformed = np.clip(b_train_transformed, 0., 1.)

rnd_idx_test = np.random.choice(g_test_data.shape[0], b_test_data.shape[0], replace=False)

b_ test_transformed = g_test_data[rnd idx test,:]

b_test_transformed = x_test_transformed + 0.5 * np.random.normal(loc=0.0, scale=1.0, size=b
test transformed.shape)

b_ test_transformed = np.clip(b_test_transformed, 0, 1)
```

Now, the transformed data set consists of white, grey and black transformed subsets. We substitute the black subset with the grey subset. Then, the substitute autoencoder maps the transformed data set to the original data set. The output will be the transformed data set where the black subset has been substituted by the grey subset.

5.4. Results

To analyze the efficacy of our model we used different flavors of autoencoders. We analyzed our model with the deep autoencoder [95], convolutional autoencoder [96], and finally analyzed with the LSTM autoencoder [97]. As our data sets are time series data, that contain repeated patterns. So we choose deep, convolutional and LSTM autoencoders. **Deep autoencoder:** We first reshaped the Skoda and the Hand Gesture data sets from 3d to 2d shape. Then, we used three layers for encoding as well as three layers for decoding. The activation function for the input and the output layers is ‘Linear’. For all hidden layers we used the Scaled exponential Linear Unit (Selu). We used the Mean Square Error (MSE) as loss function.

Convolutional Denoising Autoencoder: Here we used a 1d convolutional autoencoder. We used ‘relu’ as the activation function in all layers except the output layer. In the output layer, we applied ‘sigmoid’ as the activation function. Here, we also used MSE as the loss function.

LSTM Denoising Autoencoder: LSTM Autoencoder can learn a compressed represen- tation of sequence data and has been used on video, text, audio, and time series sequence data. Inspired by its application in time series data analysis, we implemented LSTM in our model. Here we reshaped the data into (samples, 1, windowsize$\times$features). Hence, for the left hand the data is reshaped into (samples, 1, 30×54), and for the right hand the data is reshaped into (samples, 1, 30×60) since the right hand and the left hand have 60 and 54, respectively. For the Hand gesture data set for both Subject 1 and 2 the data is reshaped into (samples, 1, 30×15).

Finally, the output of each of the different autoencoders is sent to the server for classification. The server is a machine learning platform, which contains a convolutional neural network. However, we need to check the efficiency of both the original and the transformed data set. If the transformed data set does not have the efficiency as the raw data set, then it is referred to as a misleading data set. The data set will be of no use to the server. Thus, we evaluated the performance of both the original and the transformed data sets with the convolutional autoencoder in the server.

For the Skoda data set, the results are summarized in Table 5.1, Table 5.2 and Table 5.3. Table 5.1 shows the F1-score of both the original and the transformed data set for the CNN de- noising autoencoder. Table 5.2 shows the F1-score of both the original and the transformed data set for the Deep denoising autoencoder. Table 5.3 summarizes the F1-score of the LSTM denoising autoencoder.

For the Hand Gesture data set, the results are summarized in Table 5.4, Table 5.5 and Table 5.6. Table 5.4 shows the F1-score of both the original and transformed data set for the CNN denoising autoencoder. Table 5.5 shows the F1-score of both the original and the transformed data set for the Deep denoising autoencoder. Table 5.6 summarizes the F1-score of the LSTM denoising autoencoder.

Finally, we plot the confusion matrix of different substitute autoencoders of two data sets to evaluate the performance of our model. Figure 5.3, 5.4 and 5.5 shows the transformed Skoda data sets using CNN, multilayer, and LSTM as autoencoder, respectively. The confusion matrics are for the left hand Skoda data set with Sw={4, 8, 9, 10}, Sb={1, 5, 6, 7}, and Sg={0, 2, 3}. For the Skoda data set, we can say that the LSTM substitute autoencoder performs best. We can see that the white subsets of the LSTM data set has a low false positive rate. The white subset is the desired subset, which will be used by the third party. Thus, we can use the LSTM autoencoder to hide the sensitive subsets. The other two substitute autoencoders have a very high false positive rate as compared to LSTM. Also, by looking at the confusion matrix of the LSTM and the original data set we can say that the true positive rate of the white (desired) data set is almost equal.

Table 5.1. F1-score for CNN autoEncoder of Skoda data set (OF1 stands for original data set and TF1 stands for transformed data set)

Hand	List of subsets	OF1	TF1
Left	Sw={4,8,9,10}	94.19	18.13
	Sb={1,5,6,7}	87.21	00.10
	Sg={0,2,3}	90.95	62.83
Left	Sw={0,2,3}	90.95	65.69
	Sb={1,5,6,7}	87.21	02.72
	Sg={4,8,9,10}	94.19	15.56
Left	Sw={4,8,9,10}	94.19	15.37
	Sb={1,5,6}	87.58	00.04
	Sg={0,2,3,7}	90.27	60.68
Left	Sw={4,8,9,10}	96.26	14.98
	Sb={1,5}	92.47	04.26
	Sg={0,2,3,6,7}	90.18	57.48
Right	Sw={4,8,9,10}	96.42	11.18
	Sb={1,5,6,7}	91.41	00.31
	Sg={0,2,3}	88.30	63.36
Right	Sw={1,4,10}	95.90	11.18
	Sb={2,3,8,9}	91.41	00.31
	Sg={0,5,6,7}	88.30	63.36
Right	Sw={1,4,10}	95.90	01.16
	Sb={2,3,9}	89.87	07.33
	Sg={0,5,6,7,8}	90.76	52.36
Right	Sw={4,9}	96.87	26.45
	Sb={1,2,3}	89.44	09.63
	Sg={0,5,6,7,8,10}	91.17	52.14

Table 5.2. F1-score for Deep autoEncoder Skoda data set (OF1 stands for original data set and TF1 stands for transformed data set)

Hand	List of subsets	OF1	TF1
Left	Sw={4,8,9,10}	95.47	56.54
	Sb={1,5,6,7}	87.04	00.09
	Sg={0,2,3}	87.83	66.21
Left	Sw={0,2,3}	87.83	81.78
	Sb={1,5,6,7}	87.04	00.14
	Sg={4,8,9,10}	95.47	89.51
Left	Sw={1,3,5,7}	93.50	76.37
	Sb={0,2}	88.90	12.17
	Sg={4,6,8,9,10}	94.76	83.27
Left	Sw={1,5,6,7}	87.04	13.46
	Sb={0,2,3}	87.83	52.50
	Sg={4,8,9,10}	95.47	30.18
Right	Sw={2,3,9}	89.87	01.18
	Sb={0,5,6,10}	90.19	14.28
	Sg={1,4,7,8}	93.12	09.92
Right	Sw={2,3,9,10}	97.93	06.55
	Sb={0,5,6}	91.53	63.77
	Sg={1,4,7,8}	97.90	05.48
Right	Sw={2,3,9}	89.87	82.17
	Sb={0,5,6}	89.53	01.14
	Sg={1,4,7,8,10}	94.77	90.10
Right	Sw={2,3,9,10}	97.93	19.60
	Sb={0,4,5,6}	92.56	02.28
	Sg={1,7,8}	98.12	19.40

Table 5.3. F1-score for LSTM autoEncoder Skoda data set (OF1 stands for original data set and TF1 stands for transformed data set)

Hand	**List of subsets**	**OF1**	**TF1**
Left	Sw={4,8,9,10}	95.95	93.89
	Sb={1,5,6,7}	88.00	00.04
	Sg={0,2,3}	91.89	92.05
Left	Sw={0,2,3}	91.89	91.80
	Sb={1,5,6,7}	88.00	00.26
	Sg={4,8,9,10}	95.95	93.28
Left	Sw={4,8,9,10}	95.95	94.35
	Sb={1,5,6}	87.52	00.19
	Sg={0,2,3,7}	91.49	91.13
Left	Sw={4,8,9,10}	95.95	93.91
	Sb={1,5}	92.88	00.34
	Sg={0,2,3,6,7}	89.88	88.33
Right	Sw={4,8,9,10}	96.42	94.72
	Sb={1,5,6,7}	91.41	04.14
	Sg={0,2,3}	88.33	87.55
Right	Sw={1,4,10}	95.90	95.26
	Sb={2,3,8,9}	91.37	00.05
	Sg={0,5,6,7}	89.59	89.46
Right	Sw={1,4,10}	95.90	95.55
	Sb={2,3,9}	89.87	00.02
	Sg={0,5,6,7,8}	90.76	90.58
Right	Sw={2,3,9}	89.87	86.63
	Sb={1,4}	96.41	00.66
	Sg={0,5,6,7,8,10}	91.17	89.79

Table 5.4. F1-score for CNN autoEncoder Hand Gesture data set (OF1 stands for original data set and TF1 stands for transformed data set)

Subject	List of subsets	OF1	TF1
1	Sw={1,2,3}	90.48	33.51
	Sb={0,8,9,10,11}	92.30	63.83
	Sg={4,5,6,7}	90.28	0.34
1	Sw={0,4,5,6,7}	92.63	60.45
	Sb={1,2,3}	90.40	31.98
	Sg={8,9,10,11}	91.14	1.93
1	Sw={8,9,10,11}	91.14	0.17
	Sb={1,2,3}	90.40	16.55
	Sg={0,4,5,6,7}	92.63	58.87
1	Sw={4,8,9,10}	91.14	2.73
	Sb={1,5}	91.48	45.01
	Sg={0,2,3,6,7}	92.29	55.24
2	Sw={1,2,3}	92.00	10.83
	Sb={0,8,9,10,11}	92.55	70.63
	Sg={4,5,6,7}	94.02	2.88

Figure 5.6, 5.7 and 5.8 shows the transformed Hand gesture data sets of CNN, multilayer, and LSTM as autoencoder, respectively. The confusion matrices are for subject 1 data set with Sw={8, 9, 10, 11}, Sb={1, 2, 3}, and Sg={0, 2, 4, 5, 6, 7}. For the Hand Gesture data set of Subject 1 we can say that the LSTM substitute autoencoder performs best. We can see that the white subsets of the LSTM data set has a low false positive rate. The white subset is the desired subset, which will be used by the third party. Thus, we can use the LSTM autoencoder to hide the sensitive subsets. The other two substitute autoencoders have a very high false positive rate as compared to LSTM. Also, by looking at the confusion matrix of the LSTM and the original data set we can see that the true positive rate of the white (desired) data set is almost equal.

Table 5.5. F1-score for Deep autoEncoder Hand Gesture data set (OF1 stands for original data set and TF1 stands for transformed data set)

Subject	**List of subsets**	**OF1**	**TF1**
1	Sw={1,2,3}	91.58	83.67
	Sb={0,8,9,10,11}	91.32	65.34
	Sg={4,5,6,7}	90.79	84.70
1	Sw={1,8,9,10,11}	89.37	81.78
	Sb={2,3}	92.68	01.33
	Sg={0,4,5,6,7}	91.58	89.96
1	Sw={8,9,10,11}	89.50	88.01
	Sb={1,3}	91.58	04.38
	Sg={0,2,4,5,6,7}	91.58	90.58
1	Sw={8,9,10,11}	89.50	88.80
	Sb={1,3}	91.86	0.5
	Sg={0,2,4,5,6,7}	91.58	88.80
2	Sw={8,9,10,11}	92.55	89.82
	Sb={1,3}	92.02	01.13
	Sg={0,2,4,5,6,7}	93.49	91.33
2	Sw={1,3,10,11}	92.18	88.45
	Sb={8,9}	92.52	06.76
	Sg={0,2,4,5,6,7}	93.49	92.75
2	Sw={1,8,10,11}	93.40	88.56
	Sb={2,3}	91.46	0.18
	Sg={0,4,5,6,7,9}	93.42	92.95
2	Sw={1,3,8,10}	93.6	86.22
	Sb={2,4,11}	89.24	0.74
	Sg={0,5,6,7,9}	93.73	93.47

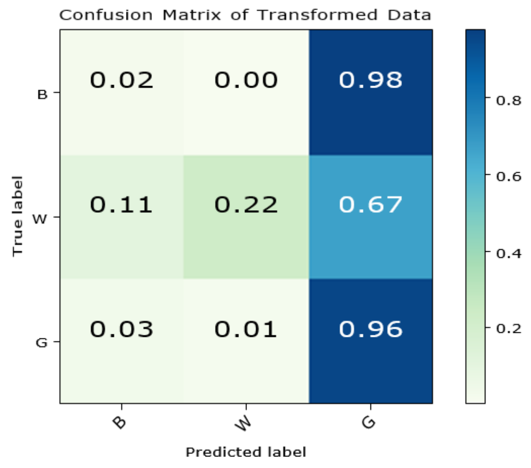

Figure 5.3. Confusion matrix of CNN autoencoder of Skoda data set

Table 5.6. F1-score for LSTM autoEncoder hand Gesture data set (OF1 stands for original data set and TF1 stands for transformed data set)

Subject	**List of subsets**	**OF1**	**TF1**
1	Sw={1,2,3}	39.98	38.78
	Sb={0,8,9,10,11}	52.52	00.37
	Sg={4,5,6,7}	81.08	80.41
1	Sw={0,4,5,6,7}	91.64	89.33
	Sb={1,2,3}	91.86	00.33
	Sg={8,9,10,11}	90.83	88.33
1	Sw={8,9,10,11}	92.04	85.99
	Sb={1,2,3}	92.06	00.21
	Sg={0,4,5,6,7}	92.00	88.77
1	Sw={8,9,10,11}	90.20	89.82
	Sb={1,3}	91.27	01.20
	Sg={0,2,4,5,6,7}	91.63	88.77
2	Sw={8,9,10,11}	92.55	89.82
	Sb={1,3}	92.02	01.13
	Sg={0,2, 4,5,6,7}	93.49	91.33
2	Sw={1,3,10,11}	93.40	88.56
	Sb={8,9}	91.46	00.18
	Sg={0,4,5,6,7,9}	93.42	92.95
2	Sw={1,3,8,10}	93.60	86.22
	Sb={2,4,11}	89.24	00.74
	Sg={0,5,6,7,9}	93.73	93.47
2	Sw={2,3,9}	89.87	86.63
	Sb={1,4}	96.41	00.66
	Sg={0,5,6,7,8,10}	91.17	89.79

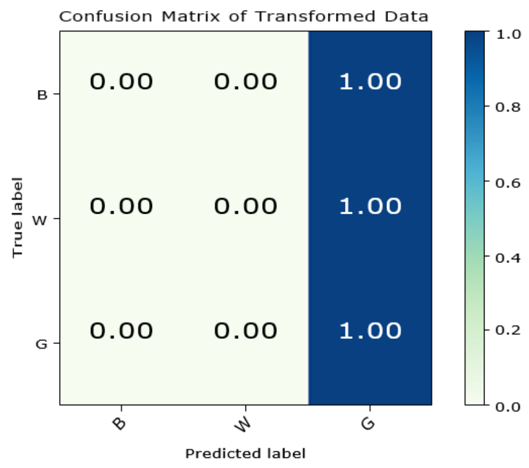

Figure 5.4. Confusion matrix of multilayer autoencoder of Skoda data set

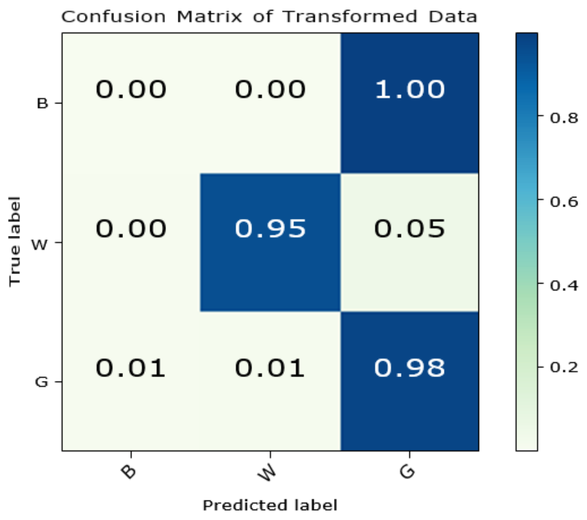

Figure 5.5. Confusion matrix of LSTM autoencoder of Skoda dataset

5.5. Security Model

As our model is based on the privacy of the sensitive data, we have to look at the security part of the model. In our security model, we first assumed that the adversaries know both the real data set and the transformed data set. When the adversary sees the transformed data set, the adversary will immediately know that it is a fake data set. Our main objective is to trick the adversary. By looking at the transformed data the adversary will not be able to distinguish if it is real or fake. To determine to what extent the original data set is indistinguishable from the transformed data set, we used a Generative Adversarial Neural Network.

For our Skoda data set model we train the GAN with the real grey subset. After training, the generator learns to produce a random data set very similar to the real one. On the other hand, the discriminator learns to distinguish between the fake and the real data set. After training, we separate the discriminator. We give the discriminator the real, transformed, and the randomly generated grey subset as input. From Figure 5.9, we can say that after 30 epochs the accuracy rate of the real and the randomly generated data set becomes almost equal. Hence, we should keep the original data set secure. If the original data set is revealed then the adversary will be able to generate a random data set, which will give the same efficiency as that of the real data set.

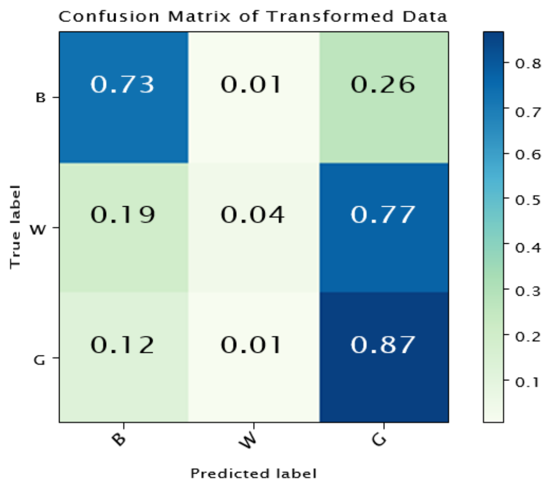

Figure 5.6. Confusion matrix of CNN autoencoder of Hand Gesture data set

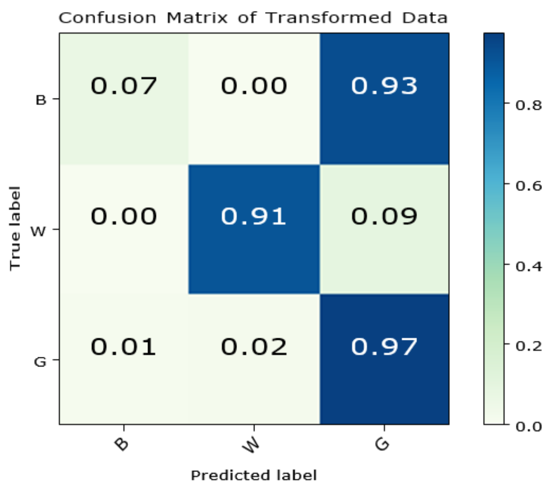

Figure 5.7. Confusion Matrix of Multilayer autoencoder of Hand Gesture Data Set

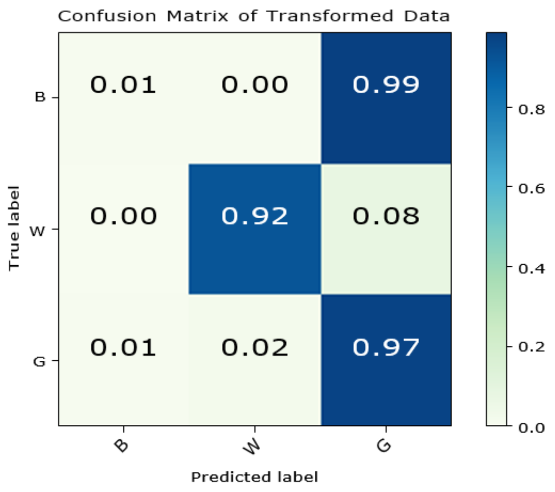

Figure 5.8. Confusion matrix of LSTM autoencoder of Hand Gesture data set

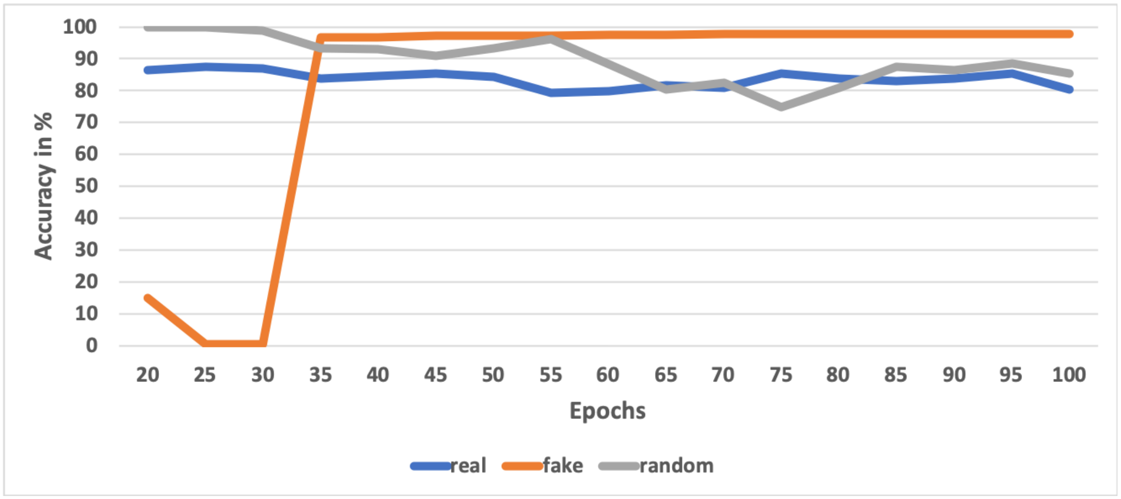

Figure 5.9. GAN accuracy is distinguishing from real, fake and randomly generated data portion

5.6. Summary

Privacy preserving of time series data is a very challenging issue especially when part of the information is private. Here, in this paper we focused on how to hide the sensitive part of time series data. We divided the data into three parts. The parts are sensitive, non-sensitive and desired. The desired part will be used for further analysis by the server. The user as well as the server is not bothered about the non-sensitive portion. At the same time, the user does not want the server to know about the sensitive portion. The sensitive portion will not be used by the server for further analysis. Thus, we substitute the sensitive portion with the non-sensitive portion. We used the property of denoising autoencoder. However, instead of only using external noise to hide data, we substitute the sensitive information with non-sensitive information. After that we add noise to the transformed sensitive information. We named the autoencoder as substitute autoencoder.

To measure the efficacy of our model, we used three different autoencoders: Deep autoen- coder, LSTM autoencoder and Convolutional autoencoder for our two different data sets. We measured the F1-score for each of the three autoencoders with both the original and the trans- formed data sets. The F1-score was also analyzed with the three different subsets. We found Skoda data set of the left hand with subsets Sw = {4, 8, 9, 10}, Sb = {1, 5, 6}, Sg = {0, 2, 3} achieved the best F1-score. By comparing the original F1-score with the transformed F1-score, the black subset has been reduced from 88% to almost 0% (0.04%), while at the same time the gray and the white subset is almost the same as the original subset. For the right hand data set with subsets Sw = {1, 4, 10}, Sb = {2, 3, 8, 9}, Sg = {0, 5, 6, 7} resulted in a good F1-score. The F1-score for the black subset has been reduced from 91.37% to almost 0% (0.04%). For the Hand gesture data set of Subject 1 with subsets Sw = {1, 2, 3}, Sb = {0, 8, 9, 10, 11}, Sg = {4, 5, 6, 7} achieved the best F1-score. By comparing the original F1-score with the transformed F1-score, the black subset has been reduced from 53% to 0.37, while at the same time the gray and the white subset is almost the same as the original subset. Subject 2 with subsets Sw = {1, 3, 10, 11}, Sb = {8, 9}, Sg ={0, 4, 5, 6, 7, 9} resulted in the best F1-score. The F1-score for the black subset has been reduced from 91.46% to 0.18%. We also plotted the confusion matrix of the three autoencoders of the left hand Skoda data set and Subject 1 of the hand gesture data set. Overall, we see that the LSTM performed better than the deep autoencoder, and the convolutional autoencoder.

We used GAN as an adversary neural network to look at the security of the skoda data set. We used the generator of the GAN to generate random noise time series data. Then, we used the random noise signal, the original gray data set, and the transformed gray data set as input to the discriminator. We can say that after 30 epochs the discriminator is not able to distinguish between the random and the real gray subsets. Hence, for security reason, the original data set should not be revealed.

6. CONCLUSION AND FUTURE WORK

In this dissertation we have used deep learning tools to achieve data privacy and security. Along with deep learning we have used different cryptographic techniques and fuzzy membership functions to achieve security. We have also shown how each of the proposed approaches is secured against different adversary attacks. To analyze the accuracy of the models we have used different optimizers, different classifiers, different noise functions, and different neural network architectures. In this dissertation we first proposed a method for both encrypting and decrypting a ciphertext. Our goal was to show how two neural networks can communicate secretly without the third party knowing about their communication. To achieve the goal, our model takes decimal values as input and converts them to ciphertext and then again back to the desired output. The proposed method shows how different network structures perform with different step values, different learning rates, and optimizers. Though the ciphertext of small length performed well, ciphertext of large length is highly recommended. The large length ciphertext will maintain the confidentiality of the secret message. The proposed method has also been proved to be secure against Brute force attack.

Secondly, we proposed a remote password authentication scheme for multiserver environments. The main idea of the proposed work is to find how a user can connect to more than one servers securely. Our proposed method has been experimented with different classifiers. The proposed method generate the keys with Diffie-Hellman key exchange protocol to make the model secure.

Thirdly, we proposed a technique that encrypts and hides sensitive information but also sends the data to different organizations securely. The main idea is to send the patient information to different organizations without compromising privacy. In order to encrypt sensitive data, our approach used three different fuzzy logic membership functions. The boundary points of the fuzzy member- ship functions are know only to sender and receiver.

Finally we proposed a different technique to hide the sensitive attributes of the data sets. Instead of perturbing the sensitive information, we substitute the sensitive information. We used a technique to substitute the sensitive attributes with non-sensitive attributes. We used two different time series data sets and measured the efficacy of our models with different flavors of autoencoders. The autoencoders which we used to measure the efficacy are LSTM, CNN and MultiLayer Autoencoder. We plotted the confusion matrix of the two data sets to evaluate the performance of our model. We evaluated the security part of the model using Generative Adversarial Neural Network (GAN). To retain the accuracy of our model we evaluated the performance of both the original and the substitute data sets using CNN as the classifier.

For the future work, various shortcomings of our proposed models should be focused on. In our first proposed approach asymmetric key encryption rather than symmetric key encryption should be used to improve the integrity and confidentiality of the data. As our first proposed approach used two neural networks, thus, the time complexity is larger in our proposed neural network model. To overcome the shortcoming, future work should also involve finding a neural network that can do both encryption and decryption. Our second proposed model used three parties to create a password secretly. The improvement to our second proposed approach could focus on a decentralized system. So that we would have only users and the servers without a trusted authority. This would make the password authentication scheme more secure and time efficient. The user should only communicate and register with the servers without sharing any legitimate information with the servers. However, the servers will authenticate the user to communicate with the servers. For our third approach we have hidden the sensitive data with Gaussian, Triangular and S-shaped membership functions. Our future work should focus on hiding the data with L- functions, R-functions and trapezoidal functions. Also, we can use different flavors of Autoencoder for sending data from the sender to different trusted organizations. Finally, more different time series data sets should be investigated using precision and F1-score as the measure. Finally, in our fourth approach we did not hide the sensitive information. Instead we substituted the sensitive information with the non-sensitive information. This technique of data transforming is very useful in activity tracking and health monitoring. In future, we should focus on extending the model without the Trusted Authority (TA). Besides applying GANs, we would like to look at other possible attacks and their appropriate responses.

REFERENCES

[1] Martin N., *How DNA Companies Like Ancestry And 23 and Me are using Genetic data*, https://www.forbes.com/sites/nicolemartin1/2018/12/05/ how-dna-companies-like-ancestry-and-23andme-are-using-your-genetic-data/ #65b51d5e6189, Last date accessed: October, 2019.

[2] Grainville K., *Facebook and Cambridge Analytica: What You Need to Know as Fallout Widens*, https://www.nytimes.com/2018/03/19/technology/ facebook-cambridge-analytica-explained.html, Last date accessed: October, 2019.

[3] Bicchierai L.F., *The 10 Biggest Revelations From Edward Snowden's Leaks* https:// mashable.com/2014/06/05/edward-snowden-revelations/, Last date accessed: October, 2019.

[4] Gressin s., *The Marriott data breach*, https://www.consumer.ftc.gov/blog/2018/12/ marriott-data-breach, Last date accessed: October, 2019.

[5] The *EU General Data Protection Regulation (GDPR) is the most important change in data privacy regulation in 20 years*, https://eugdpr.org/, Last date accessed: October, 2019.

[6] Rivest, Ronald L., Adi S., and Leonard A. *A method for obtaining digital signatures and public-key cryptosystems*. Communications of the ACM 21.2 (1978): 120-126.

[7] Arvandi, M., et al. *Symmetric cipher design using recurrent neural networks*. IJCNN'06. In- ternational Joint Conference on. IEEE, 2006.

[8] Chandra Akshay. L., Perceptron *Learning Algorithm:A Graphi cal Explanation Of Why It Works*, https://towardsdatascience.com/ perceptron-learning-algorithm-d5db0deab975, Last date accessed: October, 2019.

[9] Gupta T., *DeepLearning: Feedforward Neural Network*, https://towardsdatascience.com/ deep-learning-feedforward-neural-network-26a6705dbdc7, Last date accessed: October, 2019.

[10] Touretzky D.S., *Radial Basis Functions*, https://www.cs.cmu.edu/afs/cs/academic/ class/15883-f17/slides/rbf.pdf, Last date accessed: October, 2019.

[11] Sadeghkhani, I., Ketabi, A., Feuillet, R. *Radial basis function neural network application to power system restoration studies*. Computational Intelligence and Neuroscience, 2012.

[12] Gupta T., *Deep Learning*, https://towardsdatascience.com/ deep-learning-d5fe55326e57#.16qz1128q, Last date accessed: October, 2019.

[13] *Convolutional Neural Network* http://deeplearning.stanford.edu/tutorial/ supervised/ConvolutionalNeuralNetwork/, Last date accessed: October, 2019.

[14] *Keras LSTM tutorial – How to easily build a powerful deep learning language model*, https:// adventuresinmachinelearning.com/keras-lstm-tutorial/, Last date accessed: October, 2019.

[15] Budhiraja A., *Dropout in (Deep) Machine learning*

[16] *Early stopping*, https://metacademy.org/graphs/concepts/early_stopping, Last date ac cessed: October, 2019.

[17] Pathak H. K, Sanghi M. *Simple Three Party Key Exchange Protocol via Twin Diffie-Hellman Problem*. IJ Network Security, pp. 256-264, 2013.

[18] Pointcheval D. *Neural Networks and their Cryptographic Applications*.Pascale Charpin Ed. India, 1994.

[19] Guo, D., Lee-Ming C., and Cheng L. L. *A new symmetric probabilistic encryption scheme based on chaotic attractors of neural networks*. Applied Intelligence 10.1 (1999): 71-84.

[20] Yu, W., and Jinde C. *Cryptography based on delayed chaotic neural networks*. Physics Letters A 356.4 (2006): 333-338.

[21] Su, S., Alvin L., and Jui-Cheng Y. *Design and realization of a new chaotic neural encryp- tion/decryption network*. Circuits and Systems, 2000. IEEE APCCAS 2000. The 2000 IEEE Asia-Pacific Conference on. IEEE, 2000.

[22] Abadi, M., and David G. A. *Learning to protect communications with adversarial neural cryp- tography*. arXiv:1610.06918 (2016).

[23] Kinzel, W., and Ido K. *Neural cryptography*. Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on. Vol. 3. IEEE, 2002.

[24] Rosen-Zvi, M., et al. *Mutual learning in a tree parity machine and its application to cryptog- raphy*. Physical Review E 66.6 (2002): 066135.

[25] Klimov, A., Anton M., and Adi S. *Analysis of neural cryptography*. International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2002.

[26] Ruttor, A., et al. *Neural cryptography with feedback*. Physical Review E 69.4 (2004): 046110.

[27] Ruttor, A., et al. *Genetic attack on neural cryptography*. Physical Review E 73.3 (2006): 036121.

[28] Sarasamma, S. T., Qiuming A. Z., and Julie H. *Hierarchical Kohonenen net for anomaly detection in network security*. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35.2 (2005): 302-312.

[29] Prabakaran, N., Loganathan P., and Vivekanandan P. *Neural cryptography with multiple trans- fers functions and multiple learning rule*. International journal of soft computing 3.3 (2008): 177-181.

[30] Volna, E., et al. *Cryptography Based On Neural Network*. ECMS. 2012.

[31] Hecht-Nielsen, R. *Theory of the backpropagation neural network*. Neural Networks 1.Supplement-1 (1988): 445-448.

[32] Stallings, W., and Tahiliani M. P. *Cryptography and network security: principles and practice*. Vol. 6. London: Pearson, 2014.

[33] Micali, S., and Leonid R.. *Physically observable cryptography*. Theory of Cryptography Con- ference. Springer, Berlin, Heidelberg, 2004.

[34] Liao Y.P , Wang S.S. *A secure dynamic ID based remote user authentication scheme for multi- server environment*. Computer Standards & Interfaces, pp. 24-29, 2009.

[35] Hsiang H.C, Shih W.K. *Improvement of the secure dynamic ID based remote user authentica- tion scheme for multi-server environment*. Computer Standards & Interfaces, pp. 1118-1123, 2009.

[36] Sood S.K, Sarje A.K, Singh K. *A secure dynamic identity based authentication protocol for multi-server architecture*. Journal of Network and Computer Applications, pp. 609-618, 2011.

[37] Amin R. *Cryptanalysis and Efficient Dynamic ID Based Remote User Authentication Scheme in Multi-server Environment Using Smart Card*. IJ Network Security, pp. 172-181, 2016.

[38] Pan H.T, Pan C.S, Tsaur S.C, Hwang M.S. *Cryptanalysis of Efficient Dynamic ID Based Remote User Authentication Scheme in Multi-server Environment Using Smart Card*. Compu- tational Intelligence and Security (CIS), 12th International Conference on. IEEE, pp. 590-593, 2016.

[39] Li L.H., Lin L.C, Hwang M.S. *A remote password authentication scheme for multiserver archi- tecture using neural networks*. IEEE Transactions on Neural Networks, pp. 1498-1504, 2001.

[40] ElGamal T. *A public key cryptosystem and a signature scheme based on discrete logarithms*. IEEE transactions on information theory, pp. 469-472, 1985.

[41] Raimondo M.D, Gennaro R. *Provably secure threshold password-authenticated key exchange*. International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, pp. 507-523, 2003.

[42] Brainard J.G, Juels A., Kaliski B., Szydlo M. *A New Two-Server Approach for Authentication with Short Secrets*. USENIX Security Symposium, pp. 201-214, 2003.

[43] Obaidat M.S., Macchairolo D.T. *A multilayer neural network system for computer access se- curity*, IEEE Transactions on Systems, Man, and Cybernetics, pp. 806-813, 1994.

[44] Yang J., Park J., Lee H., Ren K., Kim K. *Mutual authentication protocol*, Workshop on RFID and lightweight crypto, 2005.

[45] Reyhani S.Z., Mahdavi M. *User authentication using neural network in smart home networks,* International Journal of Smart Home, pp. 147-154, 2007.

[46] Iuon-Chang L., Ou H.H., Hwang M.S. *A user authentication system using back-propagation network*, Neural Computing & Applications, pp. 243-249, 2005.

[47] Pattanayak S., and Ludwig S. A. *Encryption Based on Neural Cryptography*, International Conference on Health Information Science. Springer, Cham, 2017.

[48] Rosenblatt F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington DC, 1961.

[49] *Scikit-learn - Machine Learning in Python*, scikit-learn.org, Last date accessed May 2018.

[50] Bresson E., Chevassut O., Pointcheval D. *Provably authenticated group Diffie-Hellman key exchange*. Proceedings of the 8th ACM conference on Computer and Communications Security ACM, pp. 255-265, 2001.

[51] Dalenius T., and Reiss S. P. *Data-swapping: a technique for disclosure control*. Journal of statistical planning and inference, 6(1), pp.73-85,1982.

[52] Reiss S. P. *Practical data-swapping: the first steps* ACM Transactions on Database Systems (TODS), pp.68–73, 1984.

[53] Moore R. , and Richard A. *Controlled data-swapping techniques for masking public use micro- data sets* Statistical Research Division Report Series (1996): 96-04.

[54] Domingo-Ferrer J., and Mateo-Sanz J. M. *An empirical comparison of SDC methods for con- tinuous microdata in terms of information loss and disclosure risk* Proc. Joint ECE/Eurostat Work Session Stat. Data Confidentiality, Conf. Eur. Satisticians, 2001.

[55] Defays D., and Anwar M. N. *Masking microdata using micro-aggregation*. Journal of Official Statistics 14.4 (1998): 449.

[56] Domingo-Ferrer J., and Mateo-Sanz J. M. *Practical data-oriented microaggregation for statis- tical disclosure control*. IEEE Transactions on Knowledge and data Engineering 14.1 (2002): 189-201.

[57] Kim J. J., and Winkler W. E. *Masking microdata files*. Proceedings of the Survey Research Methods Section, American Statistical Association, 1995.

[58] Fuller W. *Masking procedures for microdata disclosure*. Journal of Official Statistics 9, no. 2 (1993): 383-406.

[59] Pattanayak S., and Ludwig S. A. *A secure access authentication scheme for multiserver en- vironments using neural cryptography*. Journal Of Information Assurance And Security, 13.1 (2018): 56-65.

[60] Manikandan G., Sairam N., and Priya M. Sathya *A new approach for ensuring medical data privacy using neural networks*. Biomedical Research 28.3 (2017).

[61] Kumari V. V., Rao S. S., Raju K. V. S. V. N, and Ramana K. V. *Fuzzy based approach for privacy preserving publication of data*. International Journal of Computer Science and Network Security, 8(1), 115-121 (2008).

[62] Jahan T., Narasimha G., and Rao C. V. Guru. *A Comparative Study of Data Perturbation Us- ing Fuzzy Logic to Preserve Privacy*. Networks and Communications (NetCom2013), Springer, Cham, 2014. 161-170.

[63] *Fuzzy Logic - Membership Function* https://www.tutorialspoint.cfuzzy/logicfuzzy/ logic_membership/function.htm. Last Date accessed: October, 2019.

[64] Zadeh L. A., "Fuzzy sets," Information and control 8.3 (1965): 338-353.

[65] Hinton G. E., and Salakhutdinov R. R. *Reducing the dimensionality of data with neural net- works*. science, 313(5786), 504-507, (2006).

[66] *Cervical cancer (Risk Factors) Data Set*, https://archive.ics.uci.edu/ml/datasets/ Cervical+cancer+%28Risk+Factors%29, Last date accessed: October,2019

[67] Dertat A., *Applied Deep Learning - Part 3: Autoencoders*, https://towardsdatascience. com/applied-deep-learning-part-3-autoencoders-1c083af4d798, Last date accessed: October, 2019.

[68] Baccouche M., et al., *Sequential deep learning for human action recognition* International workshop on human behavior understanding. Springer, Berlin, Heidelberg, 2011.

[69] Dai J., et al., *Towards privacy-preserving recognition of human activities*, 2015 IEEE interna- tional conference on image processing (ICIP). IEEE, 2015.

[70] CNN-wyzelab, http://www.cnn.com/2019/12/30/tech/wyze-data-breach/index.html, Last date accessed: December, 2019.

[71] Joye M., et al., *A scalable scheme for privacy-preserving aggregation of time-series data*, In- ternational Conference on Financial Cryptography and Data Security. Springer, Berlin, Hei- delberg, 2013.

[72] Dwork C., et al., *The algorithmic foundations of differential privacy*, Foundations and TrendsQR in Theoretical Computer Science 9.3-4: 211-407, 2014.

[73] Tang J., et al., *Privacy loss in apple's implementation of differential privacy on macos*, 10.12. arXiv preprint arXiv:1709.02753, 2017.

[74] Dwork C., *Differential privacy. Encyclopedia of Cryptography and Security*, pp: 338-340, 2011.

[75] Abadi M., et al., *Deep learning with differential privacy*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

[76] MNIST, http://yann.lecun.com/exdb/mnist/, Last date accessed: December 2019.

[77] CIFAR, http://www.cs.toronto.edu/~kriz/cifar.html, Last Date accessed: December 2019.

[78] Friedman J. H., *Stochastic gradient boosting. Computational statistics & data analysis*, 38.4: 367-378, 2002.

[79] Casino F., et al., *A k-anonymous approach to privacy preserving collaborative filtering*, Journal of Computer and System Sciences 81.6: 1000-1011, 2015.

[80] Xiaoyuan S. , et al., *A survey of collaborative filtering techniques*, Advances in artificial intel- ligence 2009.

[81] Hu B., et al., *Data sparsity: a key disadvantage of user-based collaborative filtering*, Asia-Pacific Web Conference. Springer, Berlin, Heidelberg, 2012.

[82] Dave S. , *Procedures for geomasking to protect patient confidentiality*, ESRI international health GIS conference, 2004.

[83] Hamton K. H. , et al., *Mapping health data: improved privacy protection with donut method geomasking*, American journal of epidemiology 172.9: 1062-1069, 2010.

[84] Zhang S. , et al., *The location swapping method for geomasking*, Cartography and Geographic Information Science 44.1: 22-34, 2017.

[85] Evfimievski A., *Randomization in privacy preserving data mining*, ACM Sigkdd Explorations Newsletter 4.2: 43-48, 2002.

[86] Oliveira S. R. M., et al., *Privacy preserving clustering by data transformation*, Journal of Information and Data Management 1.1: 37-37, 2010.

[87] Pattanayak S., et al., *Improving Data Privacy Using Fuzzy Logic and Autoencoder Neural Network*, 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2019.

[88] Malekzadeh M., et al.,*Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis*, arXiv preprint arXiv:1710.06564 2017.

[89] Pasal V., et al., *Extracting and composing robust features with denoising autoencoders*, Pro- ceedings of the 25th international conference on Machine learning. ACM, 2008.

[90] Lawrence S., et al., *Face recognition: A convolutional neural-network approach*, IEEE transac- tions on neural networks 8.1: 98-113, 1997.

[91] Kim Y., *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 2014.

[92] Goodfellow I., et al., *Generative adversarial nets. Advances in neural information processing systems*, 2014.

[93] Skoda-dataset, har-dataset.org, Last date accessed: December 2019.

[94] Bulling A., et al., *A tutorial on human activity recognition using body-worn inertial sensors*, ACM Computing Surveys, vol. 46, no. 3, pp. 33:1–33:33, 2014.

[95] Alguliyev R. M., et al.,*Privacy-preserving deep learning algorithm for big personal data anal- ysis*, Journal of Industrial Information Integration 15: 1-14, 2019.

[96] Chen M., et al., *Deep features learning for medical image analysis with convolutional autoen- coder neural network*, IEEE Transactions on Big Data, 2017.

[97] Gensler A., et al., *Deep Learning for solar power forecasting-An approach using AutoEncoder and LSTM Neural Networks*, 2016 IEEE international conference on systems, man, and cyber- netics (SMC). IEEE, 2016.