Sy Q. Truong

Researching Neural Networks and Applying for Secret Key Exchange

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

1 March 2021

Abstract

Author:	Sy Truong
Title:	Researching Neural Network and Applying for Secret Key Exchange
Number of Pages:	43 pages
Date:	1 March 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Instructors:	Janne Salonen, Head of Programme

Cryptographic practice and secret key exchange take place with a great frequency on a daily basis worldwide, which highlights their indispensable role. Therefore, researchers are increasingly concerned about security to avoid attacks, for example, man-in-the-middle attacks, one of the most common and dangerous cyberattacks.

Because the topic is important today, this study deals with secret key exchange and neural networks. The first secret key exchange protocol, the Diffie-Hellman secret key exchange protocol, has serious vulnerabilities. As a result, this study describes how the Diffie-Hellman algorithm was substituted with a more stable artificial neural network in the secret key exchange process. The neural network is distinct from the Diffie-Hellman algorithm in terms of the key exchange method and protocol. This can be seen by looking at the outcome using the Tree Parity Machines model, which is synchronizing with the synaptic weight of the neural network to generate a secret key. The secret key, in turn, is more secure than the implementation of the Diffie-Hellman secret key agreement protocol and it can be applied to the security aspects of the secret key distribution protocol, such as information confidentiality, data integrity, entity authentication and data-origin authentication.

In conclusion, the artificial neural network will significantly improve the security in the secret key exchange protocol via a public channel. The complexity of the neural network's algorithm and synchronization is more reliable than that of Diffie-Hellman in reducing the possibility of an attacker interfering with the secret key exchange process.

Keywords: Cryptography, Neural Network, Secret key exchange, TPM

Content

List of Abbreviations

List of Abbreviations

AES:	Advanced Encryption Standard
ANN:	Artificial Neuron Network
DES:	Data Encryption Standard
GPG:	GNU Privacy Guard
IDEA:	International Data Encryption Algorithm
MLP:	Multi-Layer Perceptron
NN:	Neuron Network
NIST:	National Institute of Standards and Technology
PE:	Processing Elements
PGP:	Pretty Good Privacy
RSA:	The name of an algorithm taken from 3 letters of 3 authors: Ron Rivest, Adi Shamir and Len Adleman
SSL:	Secure Sockets Layer
TPM:	Tree Parity Machines

1. Introduction

1.1. Background

As the term of cyberattack has become more and more popular, information confidentiality is not only the concern of experts in the information security field. To deal with the problem, the term of cryptography is introduced in this study as a tool to secure confidential information as it converts plaintext into cipher text, which cannot be read or understood by humans.

However, the cryptographic practice has a weakness, which is that the receiver must have the secret key of the sender in order to decrypt the ciphertext. Acknowledging this weakness, attackers target the secret key exchange process since it is one of the weakest links in the cryptographic practice.

The number and method of cyberattacks has been rising, proving that many secret key exchange protocols, for example the Diffie-Hellman protocol, have failed to provide a secure implementation. It is crucial to have a new approach in order to reduce the number of cyberattacks as well as to provide a secure protocol for the secret key exchange.

1.2. The reason of choosing the topic

The topic of the study, “researching neural network and applying to secret key exchange”, was chosen based on the insecurity of secret key exchange protocols and the new approach with artificial networks in the cryptographic process.

Cryptography offers basic services, such as the ability to send information among participants, but protection must be provided, preventing anyone from reading the information. To secure the content from the attacker, the sender encrypts the message using a symmetric and asymmetric encryption

algorithm. However, the recipient must know the sender's key to decrypt and read the message, which can be done by a key exchange protocol. Diffie-Hellman is a common key exchange protocol, which is why it is introduced in this thesis. [1.] In spite of that, the Diffie-Hellman key exchange protocol does not guarantee the confidentiality of the key exchange if the third party deliberately interferes [2.]. They can read or alter the substance of the details between the participants.

Besides, the artificial neural network having developed abilities such as learning, training, and classifying, along with diversity, chaotic value and high accuracy. It is showed the best suited for safe sharing of a secret key over public channels.

1.3. Research objective

For the sake of better protection in secret key sharing, this study describes how the Diffie-Hellman algorithm can be substituted with an artificial neural network.

The secret key is produced by the Tree Parity Machines (TPM) model synchronizing the attachment weights of the neural network. When generating a secret key, the information will be encrypted and decrypted using AES (128bit) algorithm.

1.4. Research subject and limitations

This study deals with cryptography, the concept of key exchange, and artificial neural networks; however, it is limited to applying the Perceptron neural network to secret key exchange.

2. Theoretical background of cryptography

2.1 Cryptography

2.1.1 Introduction

Cryptography is an important industry and there are many social applications. Currently, the encryption and authentication technologies are being increasingly used around the world, from security, military, and defense to civil services such as e-commerce and banking.

With the growth of computer science and the internet, the research and applications of cryptography have become increasingly complex opening numerous in-depth research directions into each application with distinctive features.

The theory of cryptography is not only used to encode and decode information but also to resolve other issues: authenticating the origin of data contents (electronic signature techniques), certifying the owner of a key (public key certificate), exchanging mechanism and safe electronic transactions. The findings of cryptographic studies have also been paired with other methods to fulfill, for instance, the complex needs of a real-world applications such as online polling applications, remote training applications, and security management applications. [3.]

2.1.2 Definition

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data-origin authentication [3].

2.2 Encryption

2.2.1 Concept of encryption and decryption

Encryption is the process of converting readable information (called plaintext) into hard information in a conventional way (called ciphertext) that is one of the techniques to secure information [4.].

Decryption is the process of converting information back from ciphertext to plaintext. An encryption or decryption algorithm is the procedure for performing encryption or decryption. [4.]

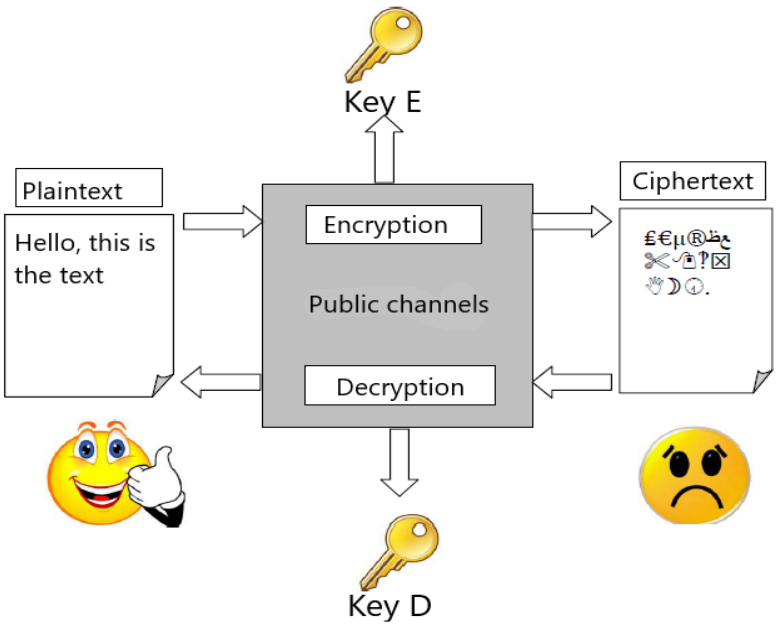

Figure 1 Encryption system diagram . Adapted from [1].

Figure 1 above illustrates the fundamental process of encryption and decryption:

Symmetric encryption: Key E = Key D

Symmetric encryption: Key E ≠ Key D

The encryption key is the value that causes the encryption algorithm to

behave in a separate way and produce its own plaintext. Usually the larger the key, the more secure the ciphertext. The range of possible key values is called key space.

An encryption system is a set of algorithms, keys to conceal information as well as clarify it.

2.2.2 Encryption techniques

A. Symmetric Encryption

Symmetric encryption is a layer of the encryption algorithm in which encryption and decryption are shared for one key (secret key) [5.].

Type of symmetric encryption algorithm:

Symmetric algorithms can be categorized into two groups, stream cipher and block cipher. Stream ciphers encrypt each bit of the letter, while block ciphers aggregate and encode multiple bits. Frames with 64-bit blocks are commonly used. The NIST-recognized AES algorithm used 128-bit blocks in December 2001. [5.]

Normally, symmetric algorithms are not used separately. The symmetric and asymmetric algorithms are used in tandem to benefit from two gains in developing the modern cryptosystem, which are, particularly, all algorithms other than SSL, PGP, or GPG. Asymmetric key algorithms are used to distribute the secret key to the higher speed symmetric algorithm.

Speed:

In general, symmetric algorithms need less calculating power than asymmetric key algorithms. In fact, an asymmetric key algorithm has a computational mass a hundred or thousand times greater than a symmetric key algorithm.

Limitation:

The limitation of symmetric key algorithms stems from the requirement for secret key distribution, each of which must contain a copy of the key. Due to the possibility that the key can be discovered by cryptographic adversary, they must often be preserved during distribution and use. The effect is loss-free enterprise that is impossible to obtain, since keys are chosen, transmitted, and processed without mistake.

To ensure safe communication for everyone in a group of *n* people, the total number of keys required is $\frac{n(n-1)}{2}$

It is, now, common practice to use slower, asymmetric algorithms to deliver the symmetric key when a transaction starts; then symmetric key algorithms take over the rest. The problem of reliably preserving key distribution also exists in the symmetry layer, but at a certain point, they can be controlled more easily. However, the symmetric keys are mostly generated locally.

Symmetric key algorithms cannot be used for authentication or non-repudiation purposes [5.].

B. Asymmetric Encryption

Asymmetric encryption is an algorithm for two separate keys, the public key, and the private key, to encode and decode [6.].

The private key for decryption and vice versa were used with a public key for encryption.

Security:

In terms of security, the asymmetric ciphers are not very different from the symmetric ciphers. There are algorithms that are widely used, mainly in theory; some algorithms are still considered safe, some have been broken. It should also be noted that widely used algorithms are not always secure. Some

algorithms have security proofs with varying standards. Many proofs associated with algorithmic breaking with well-known problems are still admitted no solution in polynomial time. Therefore, similarly to all general cryptography algorithms, the public key encryption algorithm needs to be used with caution. [6.]

Applications:

The most apparent use for public key encryption is security: only a privately owned key will decrypt a document encrypted with a user's public key.

Public key digital signature algorithms can be used for awareness. A user can encrypt text with his/her secret key. Whether another person can decrypt with the public key of the sender, the text may be assumed to come from the associated user.

Limitation:

There is a risk of anyone discovering the secret key. No asymmetric key encryption algorithms have been proved secure from attacks, depending on the statistic of the algorithm, unlike a single-pad or similar. There was no absence of some kinds of interaction between two keys, or the algorithm weakness, which permits decryption without a key or only an encryption key. The protection of the algorithm is focused on system mass estimates to overcome the relevant problems. These estimates in turn always change depending on computer capabilities and new mathematical discoveries. [6.]

The capacity of a man-in-the-middle attack: the intruder takes advantage of the public key delivery to change the public key. Once the public key is forged, the intruder will get packets in the middle and decode them and then encrypt them with the right key and send them to the recipient so that detection is not feasible. The protected key exchange methods for the identification of transmitters and the integrity of information will avoid such attacks.

Computational volume:

Achieving the equivalent security requires significantly more computational volume than the symmetric cipher algorithm. Thus, both types of algorithms are often used to achieve high performance, complementing each other. This model produces a symmetric key for the interaction by an exchange participant. The key is shared safely by an asymmetric scheme of key encryption. Then all parties share information on secrecy using a symmetric encryption scheme. [6.]

2.3 Key exchange

2.3.1 Introduction of Diffie-Hellman key exchange

The first invented key exchanging system of cryptography is Diffie-Hellman key exchange. The key exchange system enables both parties (the person and the communication entity) to generate a common secret key to encrypt data used on unsecured communication networks, without prior agreement between the two parties on a secret key. The secret key created will be used in the symmetric key encryption process to encrypt the data. [1.]

In 1976, Whitfield Diffie and Martin Hellman first presented their protocol. In 2002, Hellman proposed the algorithm to be called Diffie-Hellman-Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public key cryptography. [1.]

2.3.2 Diffie-Hellman key exchange protocol

Diffie Hellman established public secrecy to be used for secure data exchange over an insecure public communication channel. According to Singh [4], figure 2 below represents the basic scheme of the key exchange through a color paint example. The key point of this idea is that Alice and Bob exchange the secret paint color through the paint mix.

Firstly, Alice and Bob blend in their own private paint with yellow shade.

Then, each person transfers their mix to the other via a public transit channel.

Upon receiving the other's mix, each will mix more with their own secret color and get the final mix.

The final paint mix is the same for both people and only the two know. The key here is that for an outsider it will be difficult (in terms of calculation) to figure out the two's shared secrets (that is, the final mix). This shared secret will be used by Alice and Bob to encrypt and decrypt data on the public networks. It is important to notice that the first color (yellow) is optional; however, it was decided between Alice and Bob in advance. This paint color is often believed to be third party proprietary without exposing Alice's and Bob's ultimate mutual secrets.

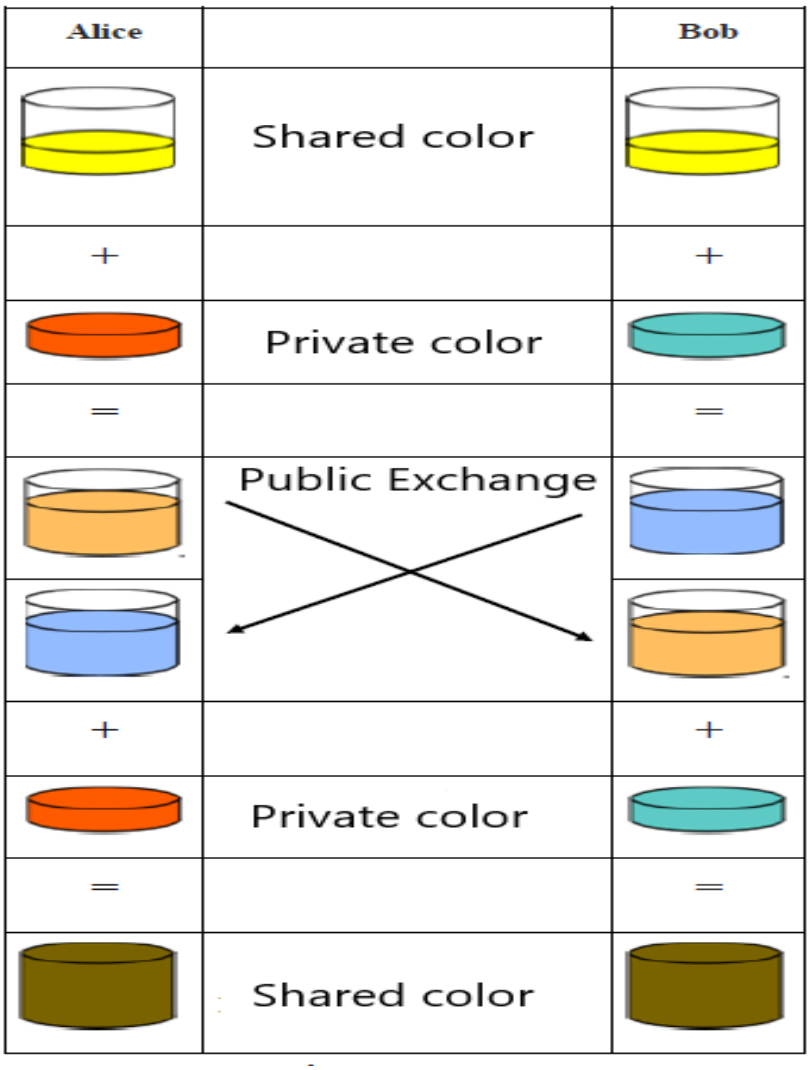

Figure 2. Alice and Bob's secret paint color exchange. Adapted from [4]

The protocol is interpreted mathematically, indicated in table 1 as follows:

The protocol uses integer multiplication module *p*, where *p* is the prime, with *g* being the primitive root mod *p*. The disclose part is written in blue in the following case, and the secret is written in red.

Table 1 The mathematical protocol for sharing secrets between Alice and Bob [4]

Alice				Bob		
Secret	Public	Calculate	Send	Calculate	Public	Secret
A	p.g		p,g →			B
A	p.g.A	$g^a \bmod p = A$	A→		p.g	B
A	p.g.A		←B	$g^b \bmod$	p.g.A.B	B
a,s	p.g.A.B	$B^a \bmod p = s$		$A^b \bmod$	p.g.A.B	b,s

The prime number p =23 and primitive g =5 is decided by Alice and Bob.

1) Alice gives Bob the value $A = g^a \bmod p$ to pick an integer secret of a = 6.
- $A = 5^6 \bmod 23$
- $A = 15{,}625 \bmod 23$
- $A = 8$

2) Bob gives Alice the value $B = g^b \bmod p$ to pick an integer secret of b = 15.
- $B = 5^{15} \bmod 23$
- $B = 30{,}517{,}578{,}125 \bmod 23$
- $B = 19$

3) Alice calculates $s = B^a \bmod p$
- $s = 19^6 \bmod 23$
- $s = 47{,}045{,}881 \bmod 23$
- $s = 2$

4) Bob calculates $s = A^b \bmod p$
- $s = 8^{15} \bmod 23$
- $s = 35{,}184{,}372{,}088{,}832 \bmod 23$
- $s = 2$

5) Alice and Bob share the secret number 2 since 6*15 corresponds to 15*6.

Both Alice and Bob get the final common value because $(g^a)^b = (g^b)^a$ mod *p.* It is important to note that only *a*, *b* and $g^{ab} = g^{ba}$ mod *p* is kept secret. All other values such as *p*, *g*, g^a mod *p* and g^b mod *p* are transmitted publicly. If the shared secrets have been determined, Alice and Bob can only transmit data to open media by two people as a standard encryption key. In practice, for the protocol to be secure, a much larger value for *a, b,* and *p* is used, since in the above example, for n mod 23, it is just 23 different outcomes (hence the attacker only needs to try 23 cases to find the secret key). If prime *p* has a minimum of 300 digits, *a* and *b* have 100 or more digits. Even modern computers cannot find *a* knowing *g*, *p*, g^b mod *p* and g^a mod *p*. [4]

The number *g* is not necessarily a great root. In practice, the values 2, 3 and 5 are usually used.

2.3.3 Limitation

The key exchange of Diffie-Hellman is vulnerable to a man-in-the-middle attack. To explain, a competitor called Eve intercepts Alice's transmitting value as Alice transmits a public key to Bob and sends her own values to Bob. Eve substitutes the value of Bob with her own value and gives it to Alice as Bob transfers the public key onto Alice. Eve and Alice then settle a common key, and Eve and Bob agree on a separate common key. After this exchange, Eve will simply decode and read all the messages sent by Alice and/or Bob before re-encrypting with the correct key and sending it to the other side. [2; 7.]

3. Theoretical background of the artificial neural network

3.1 Development history of neural network

Studies of the human brain have been conducted for thousands of years. With the development of science and technology, it is completely natural that humans have begun to study artificial neurons. In 1943, the neuroscientist

Warren McColloch and the mathematician Walter Pitts wrote a paper explaining the functioning of neurons [8], the first occurrence marking the birth of an artificial neural network. They also developed a basic neural network with electric circuits. Their neurons are used as a fixed-threshold binary device. The observations are simple logical functions as “a OR b” or “AND b”.

Following these studies, in 1949 Donald Hebb published a book called Organization of Behavior. The book has shown that artificial neurons become more efficient each time they are used.

The advances of the computers in the early 1950s made it possible to model the principles of theories about how humans think. After many years at IBM research laboratories, Nathaniel Rochester made his first attempt to simulate a neural network [9].

In this period, traditional computation has achieved great success, meanwhile neuron research was in its infancy. The supporters, of “think machines” ideology, are still maintaining their stance. In 1956, the artificial intelligence experiment in Dartmouth started a new age in both artificial and neural intelligence areas. Its positive effect is to boost further the interest of scientists in artificial intelligence and the simple neural network processing in human brain. [10.]

In the following years of the Dartmouth project, John von Neumann suggested to use either voltage relays or vacuum lamps to mimic basic neurons. [9]

Neuroscientist Frank Rosenblatt also began work on Perceptron. After this period of research, Perceptron has become the oldest-fashioned neural network in operation and built-in computer hardware. The single-tier Perceptron is useful in classifying one of two groups for a sequential set of important inputs. The Perceptron measures the weighted sum of inputs, then extracts this sum for a threshold and gives one of the two values desired. However, Perceptron has many limitations, described in the 1969 book on Perceptron by Marvin Minsky and Seymour Papert. [11]

The ADALINE (ADAptive LINear Elements) and MADALINE model (Multiple ADAptive LINear Elements) were designed in 1959 by Bernard Widrow and Marcian Hoff of Stanford University. These models use the Least-Mean-Squares (LMS: Minimum squared average) rule. MADALINE is the first neural network used for addressing a specific issue [9]. It is an adaptive filter that eliminates the echo on the phone line. Today, this neural network is still used in commercial applications.

In 1974, Paul Werbos developed and applied a backpropagation method [12]. However, it took a few years for this method to become popular. The back-propagation LAN networks are most recognized and widely accepted today.

Unfortunately, these early successes have led people to overthink the possibilities of neural networks. It is the exaggeration that had a negative effect on the development of science and technology of today as people feared that it is the time for machines to do all the human things. These concerns have caused people to begin to oppose research on neural networks. This lull period lasted until 1981.

In 1982, in an article submitted to the National Academy of Science, John Hopfield showed, through simple and logical mathematical analysis, how neural networks functioned and what they can do [13]. The contribution of Hopfield lies not only in the importance of science research, but also in encouraging research into neural networks.

A conference on neural network collaboration and competitiveness in Kyoto, Japan was also held during this period, in which the U.S and Japan participated. After the conference, Japan announced their efforts in creating 5th generation computers. Receiving this, the U.S newspapers expressed concern that in this region, they could fall behind. The United States quickly expanded support for research and implementations for the neural network soon afterwards.[13]

Annual meetings on Neural Networks for Computing were conducted by the American Institute of Physics in 1985.

Today, studies of the application of neural networks in order to resolve practical problems are not just in theoretical science. Neural network technologies are rapidly and thoroughly developed. Typical fields include language processing, character recognition, voice recognition, pattern recognition, signal processing and data filtering.

3.2 Concepts of neural network

3.2.1. Study about neurons

A. Biological neurons

Through the study of the brain, it was found that the human brain consists of about 10^{11} neurons involved in about 10^{15} connections on the transmission path. Each of these lines is about one meter long. Neurons have many features in common with cells in the human body. In addition, they have abilities that cells do not have, namely the ability to receive, process, and transmit electrochemical signals on neural pathways, which form the communication system of the brain. [14.]

Biological neurons are composed of the following main components: soma, dendrites, and axons as figure 3 below illustrates:

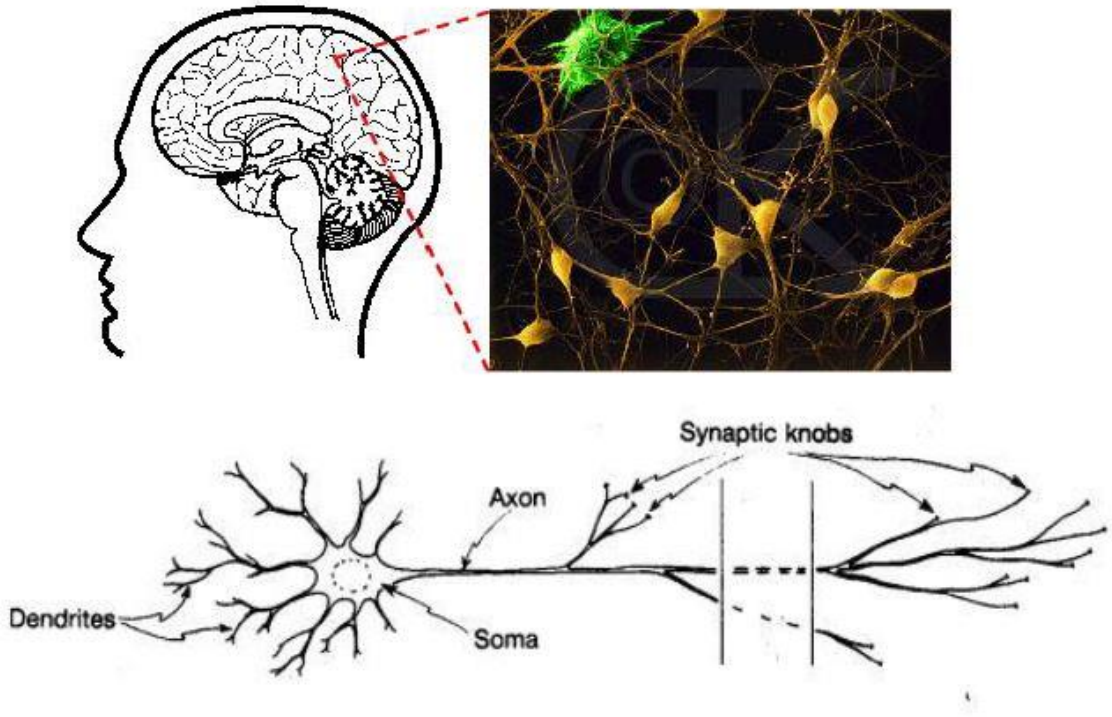

Figure 3 Biological neuron model. Copied from [14]

Soma is the body of a neuron.

Dendrites are thin, long strings linked to the soma that transmit data (in the form of voltage impulses) to the soma for processing. The data are aggregated inside the soma.

Another type of somatic signaling fibers are axons. Unlike the dendrites, axon can generate voltage impulses, which are the signal wires going from neurons to other locations. Only when the potential in the soma exceeds a certain threshold value, the axons generate an electrical impulse; otherwise, they are at rest.

The axons connect to the dendrites of another neuron through so-called synaptic knobs. As the potential of synaptic knobs is increased due to the impulses emitted by the axons, the synaptic knobs produce certain chemicals that will open the ‘door’ on the dendrites for the ions to pass [14]. It is this current of ions that alters the potentials on the dendrites, creating data that broadens to other neurons.

The neuron activity can be summarized accordingly: the neuron sums all the input potentials it receives and then emits a potential impulse if the sum is greater than a certain threshold. The neurons are connected by the synaptic knobs. Synaptic knobs are known to be powerful as they allow signals to be transmitted easily to other neurons. In contrast, weak synaptic knobs will make signal transmission tremendously difficult.

The synaptic knobs play an important role in learning. As people study, the activity of the synaptic knobs intensifies, creating strong bonds between neurons. It can be said that the more a person learns, the more synaptic knobs that one has, and that the synaptic knobs are strong. In other words, the more connections between neurons, the more sensitive the neurons.

Based on knowledge of biological neurons, people build artificial neurons in the hopes of creating a model with the same power as the brain

B. Artificial neural network

An artificial neural network can be understood as follows:

> A computation unit that has many inputs and one output, each input coming from a connection. A neuron characteristic is a non-linear activation function that converts the linear combination of all input signals into output signals. This activation function ensures nonlinearity for neural network computation. A neuron is an information processing unit and is the rudimentary component of a neural network. [15.]

Figure 4 below describes the model of a neural network:

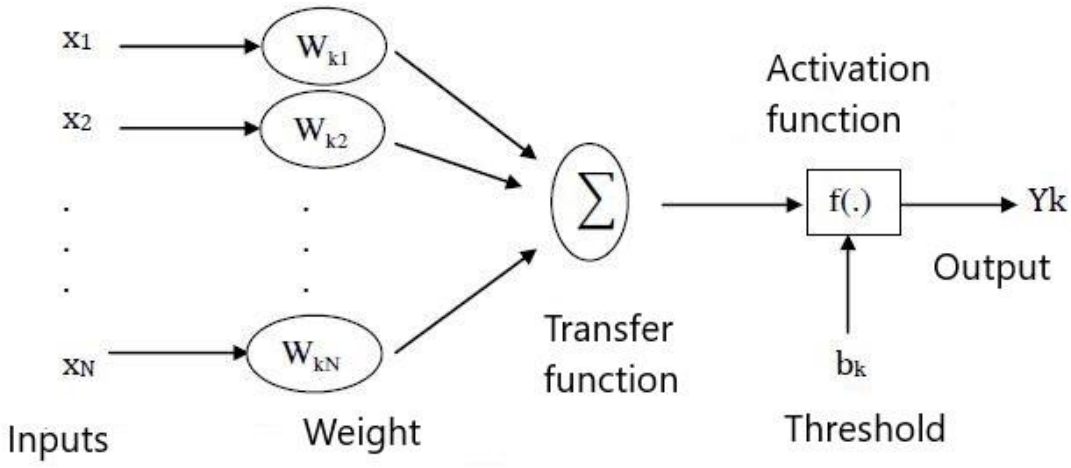

Figure 4 Artificial neural network model. Copied from [14]

According to Sharma [14], the basic components of an artificial neuron include:

- Set of inputs: Input signals of neurons are usually given in the form of an N-dimensional vector.
- Set of links: Each link is represented by a weight (call weight link). The weight connects input signals *j* to neuron *k* which is often denoted w_{kj}. Usually, these weights are randomly generated at network initialization and are continually updated during training.
- Transfer function (summing function): Usually used to sum the product of the input by its associated weight.
- Threshold (bias): This threshold is usually included as part of the activation function.
- Activation function: This function is used to limit the output range of each neuron. It takes the input as the result of the transfer function and the given threshold. Each neuron's output is often limited in segment [0,1] or [-1, 1]. The activation functions are diverse, and they can be linear or non-linear functions. The selection of the activation function depends on each problem and the experience of the network designer.
- Outputs: It is the output signal of a neuron, with each neuron having at most one output.

The neural network can be activated using the functions presented in table 2:

Table 2 Basic activation functions in neural network

Function name	Formular
Hardlim	a = 0 if n < 0 a = 1 if n ≥ 0
Hardlims (SIGN)	a = -1 if n < 0 a = 1 if n ≥ 0
Purelin	a = n
Satlin	a = 0 if n < 0 a = n if 0 ≤ n ≤ 1 a = 1 if n > 1
Satlins	a = -1 if n < 0 a = n if 0 ≤ n ≤ 1 a = 1 if n > 1
Tansig	$a = \frac{e^n - e^{-n}}{1 + e^{-n}}$
Poslin	a = 0 if n < 0 a = 1 if n ≥ 0
Compet	a = 1 at neuron that has biggest n a = 0 with the rest
Logsig	$a = \frac{1}{1 + e^{-n}}$

3.2.2. Artificial neural network

Artificial neuron network (ANN), neural network for short, is a model of information processing simulating the way information is processed by biological neuron systems. It is made up of a large number of elements (called processing elements or neurons) interconnected through synaptic links (called weights) that work as a unified body to solve one specific problem. [14.]

An artificial neural network is configured for a specific application (such as pattern recognition and data classification) via a training process from a set of patterns. In essence, it is the process of adjusting the bonding weights between neurons. It is the system consisting of simple processing elements, like the neurons of the human brain, operating in parallel and connected by neuronal connections. Each connection has a weight.

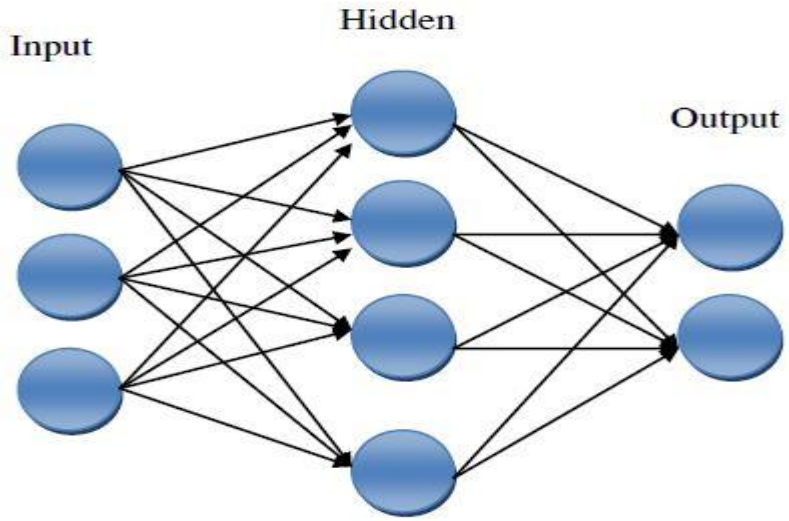

Figure 5 A simple model of an ANN. Copied from [14]

The ANN model, as presented in figure 5 above, includes three layers: input, hidden and output. Each node in the input layer takes a value of an independent variable and passes it to the network layer.

3.3 Neural network features

3.3.1. Nonlinearity:

A neuron can compute linearly or non-linearly. A neural network, which is made up of a connection of non-linear neurons, will itself be non-linear. Especially, this nonlinearity is distributed across the network [14]. Nonlinearity is an extremely important property, especially when the physical mechanisms that generate input signals (for example, voice signals) are inherently nonlinear.

3.3.2. Input-output correspondence:

Although the concepts of learning and training have not been discussed, in order to understand the input-output relationship of the neural network, these concepts will briefly be introduced as they are related to a common learning model called learning with a supervised tutor. This learning model updates the synaptic weight value in in order to produce the actual output which is as close as possible to the desired output on the input.

Each cumulate example includes an input signal and a desired output, respectively. The neural network receives an example taken at random from the above set at its input, and the synaptic weights of the network are altered so that the difference between the desired output and the actual output of the network can be minimized according to an appropriate statistical standard. Network accumulation is repeated with instances in aggregation until the network reaches a steady-state network where there is no significant change in weights. The pre-applied cumulative examples can be re-applied during the cumulative session but in a different order. Thus, neural networks learn from examples by constructing input-output correspondence for the problem that needs to be solved. [14.]

3.3.3. Adaptation

Neural networks have the default ability to change the weights according to changes in their surroundings [14]. In particular, a neural network that has been accrued to operate in a given medium can easily accumulate when there are slight changes in the operating environment conditions.

3.3.4. Evidence for solution

In the context of sample classification, a neural network can be designed to give information not only about the classified sample, but also about the reliability of the decision made. This information can be used to eliminate ambiguous or ambiguous patterns.

3.3.5. Error Acceptance

A neural network, installed in the form of hardware, is inherently error tolerant, or raw computational, in the sense that its performance deteriorates only when there are adverse operating conditions. For example, if a neuron or its interconnections fail, the re-identification of a stored pattern deteriorates in quality.

3.3.6. Ability to install VLSI (Very-large-scale-integrated)

The massive parallel nature of a neural network makes it extremely fast in computing for some tasks. This feature also makes a neural network suitable for installations using the very-large-scale-integrated (VSLI) technique. This technique allows the construction of large scale parallel computational hard circuits. That is why the outstanding advantage of VLSI is that it provides an efficient means to handle highly complex behavior. [16.]

3.3.7. Similarities in analysis and design

Essentially, neural networks have the same properties as information processors. This is raised with the same meaning for all areas related to neural network application. This feature is shown in a number of points, which are discussed below:

Firstly, the neurons, in one form or another, represent a prevalent component for all neural networks. Second, this unity makes it possible to apportion learning theories and algorithms in many different applications of neural networks. Finally, a modular network can be built utilizing an integration of different models. [14.]

3.4 Classification of artificial neural networks

Although every single neuron can perform certain information-processing functions, the power of neuromodulation is mainly obtained by combining neurons in unified architecture [14]. A neural network is a computational model defined via parameters: neural type (as a node if the whole neural network is considered as a graph), connection architecture (the organization of the connection between neurons) and learning algorithm (algorithm to train the network). In essence, a neural network has the same functions as the mapping function F has: $X \rightarrow Y$, where X is the input state space and Y is the output state space of the network. The networks simply map the input vectors $x \in X$ to the output vectors $y \in Y$ through a “filter” of weights. That is, $y = F(x) = s(W, x)$, where W is the associative weight matrix. Network operations are often the computation of a real number on matrices.

3.4.1 Types of neural network models

The way neurons in the network are connected determines the architecture of the network. Neurons in the network are fully interconnected, meaning each neuron is either connected to all other neurons, or locally connected, for

example, to neurons in different layers. There are two main types of network models:

The auto-associative network is a network with input and output neurons that are identical [17], as described in figure 6. The Hopfield network is a type of this self-matching network.

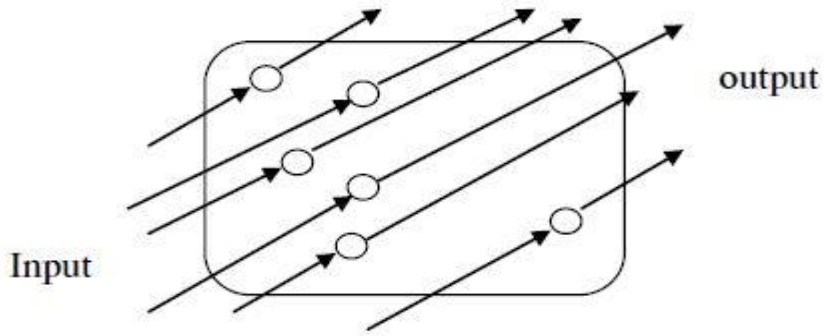

Figure 6 Auto-associate network. Copied from [17]

The hetero-associative network is a network with a separate set of input and output neurons [18], as demonstrated in figure 7. Perceptron, the multiplayer Perceptron network (MLP) and the Korhonen networks fall into this category.

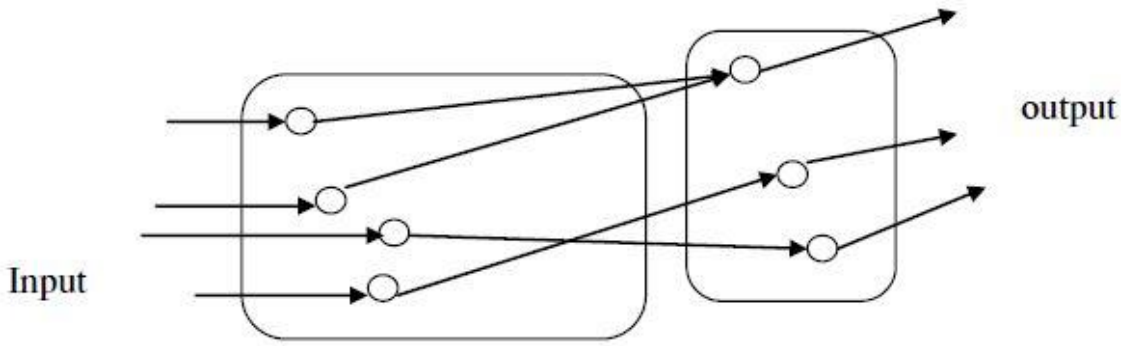

Figure 7 Hetero-associate network. Copied from [18]

In addition, depending on whether the network has backward connections (feedback connections) from the output neurons to the input neurons, it is divided into two types of network architecture:

Feed-forward architecture is a type of network architecture, as figure 8 below, without reverse connections from the output neurons to the input neurons [19]; the network does not record previous output values and neuron activation states. Straight transmission neural networks allow signals to travel in a single path; from the input to the output, the output of any layer will not affect that layer. Perceptron networks are straight networks.

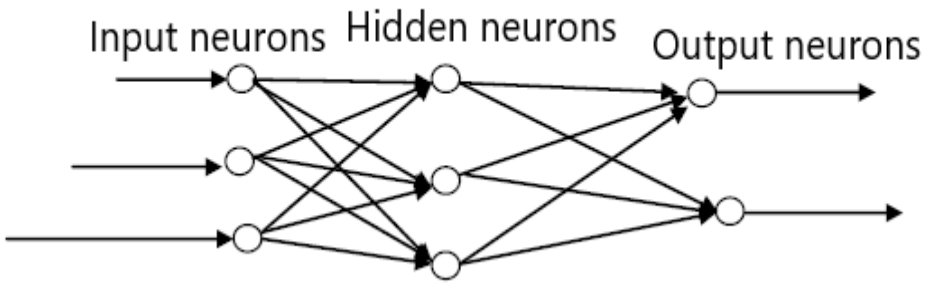

Figure 8 Straight transmission network. Copied from [19]

Feedback architecture is the network architecture type that has connections between the output neuron and the input neuron. The network stores the previous output, and the next state depends on not only the input, but also on the previous output that has been stored by the network [20]. This category includes the Hopfield network. The following figure illustrates this architecture.

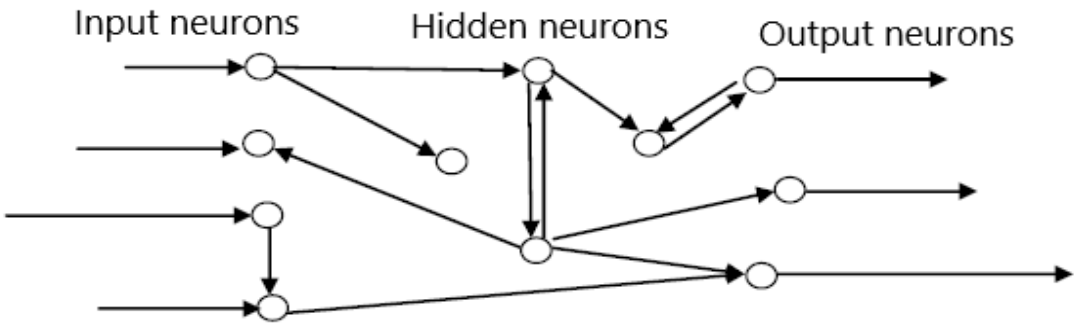

Figure 9 Feedback architecture network. Copied from [20]

3.4.2 Perceptron neural network

Perceptron is the simplest neural network. It consists of only one neuron, taking input as a vector the components, of which are real numbers. The output is either +1 or -1.[21]

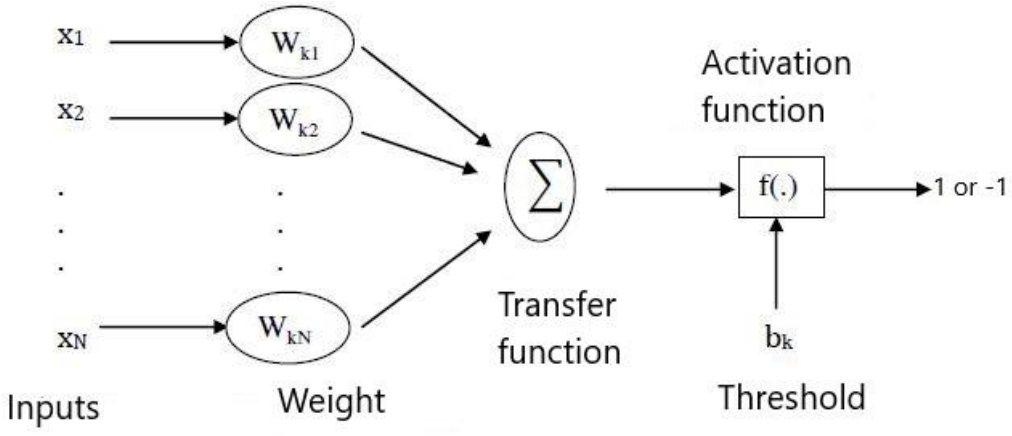

Figure 10 Perceptron network. Copied from [21]

Figure 10 above demonstrates that the output of the network is determined as the network takes the weighted sum of the components of the input vectors. This result with the threshold *b* is passed into the transfer function (Perceptron uses the Hard-limit function as the transfer function) and the result of the transfer function will be the output of the network.

Perceptron allows for precise classifications in cases (patterns are on the opposite sides of an ultra-flat). It also correctly categorizes the output AND and OR functions and functions of the correct form when n in its m inputs is true ($n \leq m$). Perceptron cannot classify the output of the XOR function.[21]

3.4.3 Multilayer Linear Network (Multilayer Perceptron)

The most widely used neural network model is the multilayer linear network model (Multilayer Perceptron, MLP). A general MLP network, showed in figure 11, is the one with the n ($n \geq 2$) layer (the input layer is usually not

considered), which includes an output layer (n^{th} layer) and the (n-1) hidden layer.[22]

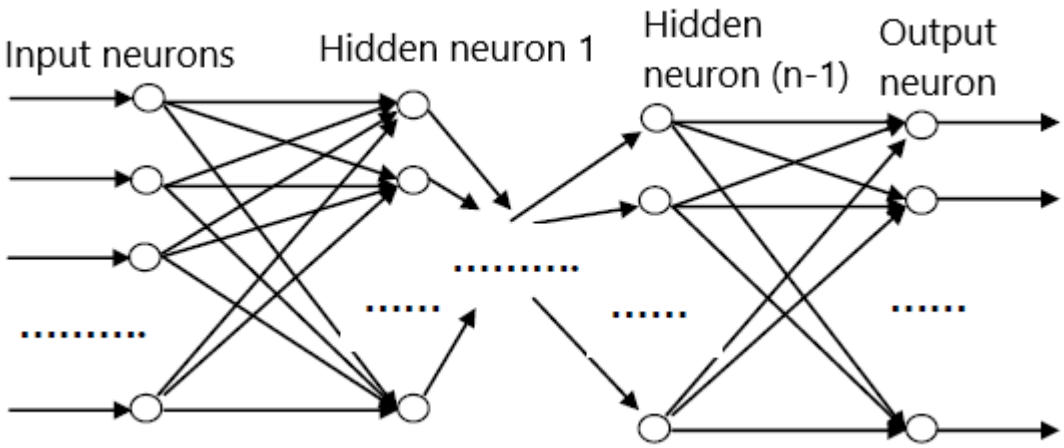

Figure 11 General MLP network. Copied from [22]

According to Hastie [22], the architecture of a general MLP network can be described as follows:

- Inputs are vectors (x_1, x_2, …, x_p) in a p-dimensional space.
- Outputs are vectors (y_1, y_2, …, y_q) in q-dimensional space. For classification problems, p is the size of the input sample, and *q* is the number of the classes to be classified.
- Each neuron in the posterior layer is associated with all neurons in its preceding layer.
- The output of the anterior neuron is the input of the neuron in the layer following it.
- The operation of the MLP network: at the input layer, the neurons receive the input signal to process (calculate the weight and send to the transfer function) and then output the result (which is the result of the transfer function). This result, then, will be transmitted to neurons in the first hidden layer; the neurons here receive a number, called X, as input signals and process and send the results to the 2^{nd} hidden layer. The processes continue until the output neurons give a result.
- There are proven results as listed below:

- Any Boolean function can be represented by a two-tier MLP network in which neurons use the sigmoid transfer function.
- All continuity functions can be approximated by a 2-layer MLP network using a sigmoid transfer function for latent layer neurons and a linear transfer function for output neurons with arbitrarily small errors.
- Any function can be approximated by a 3-layer MLP network using a sigmoid transfer function for latent layer neurons and a linear transfer function for output neurons.

3.5 Build a neural network

Basically, a neural network can be understood as a directed graph as shown in figure 12, where the vertices of the graph are neurons, and the edges of the graph are the connections between neurons.

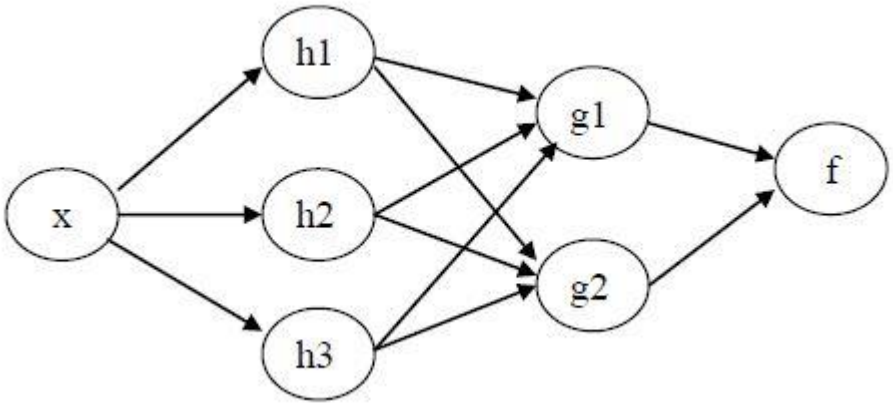

Figure 12 Simple directed graph

In order to build a neural network, a directed graph is built first: the number of vertices of the graph is equal to the number of neurons in the network, and the value of the edges is the synaptic weight.

3.6 Neural network training

Machine training is a process of changing the behavior of objects in a way that enables them to perform better in the future.
A neural network is trained so that with a set of X input vectors, the network is capable of producing its desired set of output vectors Y. The X set used for network training is called the training set. The x elements in X are called training examples. Essentially, the training process is the change of the weight of the network. During this process, the network weights will converge gradually to values so that for each input vector x from the training set, the network will produce the desired output vector.

There are three popular training methods: supervised training, unsupervised training, and reinforcement training.

3.6.1 Supervised training

Supervised training uses a set of pair inputs and desired outputs [23]. It is a training process with the supervision of a “teacher”. Similarly, teaching a child letters, the teacher gives out an “a” and tells the child this is “a”. This is done on all letter patterns. Then when testing, the teacher gives out any letter (can be written slightly differently) and asks the child the given world.

For supervised training, the training set is given as:
D = {(x,t) | (x,t) ∈ [IR^N x R^K]} where: x = (x_1, x_2, …, x_N) is the N-dimensional feature vector of the training set and t = (t_1, t_2, …, t_K) are the respective K-dimensional target vectors. The task of the algorithm is to establish some kind of network computation so that for each input feature vector, the error between the actual output value and the corresponding target value is the smallest. For example, a network can learn to approximate a function t=f(x) representing the relationship of a set of training patterns (x,t). [23]

Thus, with supervised training, the number of classes to be classified is known in advance. The task of the algorithm is to define a classification for each input vector which is correctly classified into its class.

3.6.2 Unsupervised training

Unsupervised training can be understood as training that does not need any supervision.

In the unsupervised training problem, the training data set is given as follows: D = {(x_1, x_2, …, x_N)}, where (x_1, x_2, …, x_N) is the characteristic vector of the training pattern. The task of the algorithm is to divide the D data set into subgroups, each containing the same input vectors.[23]

Thus, with unsupervised training, the number of classifications is not known in advance, which depends on the criteria for evaluating the similarity between samples. As a result, there might be different classifications.

3.6.3 Reinforcement training

Reinforcement training is, simply, the combination of both above models. This method is as follows: for the input vector, it is important to observe the output vector calculated by the network. If the results are considered good, then the network will be rewarded in the sense of increasing the weights; otherwise, the network will be penalized, and the irrelevant connection weight will be reduced. Hence, reinforcement training is critical training, opposed to supervised training that is teacher training.[24]

3.7 Knowledge representation for neural network

Knowledge is stored information or models used by people and machines to represent the real world, make judgements about the world, and have responses consistent with the real world. Knowledge includes facts and laws.

The basic characteristics of the knowledge representation are:

- What information is actually represented?
- How is the information physically encoded for later use? In practical applications of intelligent computers, it can be said that a good solution depends on a good knowledge variable. The same is true for neural networks, a special class of intelligent machines. However, the possible representations from the inputs to the internal parameters of the network are varied, and the tendency is that finding a suitable solution to represent knowledge by means of a neural network becomes a design challenge.
- It should be emphasized here that the neural network stores information about the real world by its own structure, both in terms of shape and internal parameter values (which can be changed for new capture). The primary task of neural networks is learning a real-world model to achieve certain goals of interest.

Because the structure of a neural network is extremely diverse, to be able to represent knowledge effectively, the four following general rules should be complied with:

Rule 1: The analog-layer inputs should always produce similar representations in the network and should be classified of the same type. In this standard, several measures are often used to determine the “similarity” between inputs (for example, Euclidean distance).

Rule 2: Molecules that can be divided into separate layers should have very different representations in the network.

Rule 3: If a feature is particularly important, there should be a large number of neurons involved in the representation of this feature in the network. The large number of neurons ensures a high degree of accuracy in decision making and improves tolerance to broken neurons.

Rule 4: The initial information and immutable properties should be included in the original design of the neural network, and as such will reduce the burden on the training process. Rule 4 is especially important because if being applied appropriately, it leads to the ability to create neural networks with a specific architecture. This is a real concern for a number of reasons:

- Neural networks of vision and biological hearing are known to be very specialized.
- A neural network with a specialized structure usually has a small number of free parameters suitable for adjustment rather than a fully connected network. Thus, specialized neural networks need a smaller data set for accumulation, and they learn faster and are often more likely to generalize.
- The speed of transferring information over a dedicated network is faster.
- The cost of building a dedicated network will be smaller due to its small size compared to a fully connected network.

3.8 Problems of neural networks

When building a neural network application, the following issues need to be considered:

- The issue of neural network architecture is the choice of neural network models. It will depend on the data and application presentation. The excessively complex models lead to problems with choosing the training process or choosing a learning algorithm.
- Choosing a learning algorithm is the challenge of the balance between learning techniques. Nearly any algorithm will accomplish the goal with the precision of the metrics for training on pre-fixed datasets. However, the choice and navigation of the algorithm for training on these datasets require a considerable number of experimentations, which is vastly

important. On a model, if choosing a suitable algorithm and evaluation function, the neural network can give great results.

The usage of neural network can be instructed as:

Building an initialization network (using a hidden layer with a number of neurons = ½ of the total number of neurons of the input and output layers).

Train the network uses the learning algorithm. It should be done on different networks to avoid local minima.

If the machine “does not remember”, then it is necessary to add some neurons to the hidden layer.

Conversely, if the machine learns as “rote learning”, then it is necessary to remove a few neurons from the hidden layer.

Once a relatively good network architecture is found, it is necessary to resample the data set and retrain it to find new networks.

3.9 Neural network applications

Many people in the past few years have been interested in neural networks which have successfully been applied in many different fields, such as finance, medicine, geology and physics. Indeed, wherever there are problems related to prediction, classification and control, neural networks can be applied. For example, the ability to identify human faces in information management systems which is related to people. In addition, they can be used in human resource management and criminal sciences.

Combined with open logic, neural networks have revolutionized the intelligentization and universalization of high-tech controllers both now and in the future, for instance, the application of automatic train steering system control and incident forecast system.

4. Architecture overview and implementation of key generation

4.1 Architecture overview and algorithm:

The idea is to build a Perceptron neural network in which the synaptic weights elements are synchronized into this network layer and in which the synaptic weights will be the secret keys in the Tree Parity Machines model.

The Tree Parity Machines (TPM) model consists of an input vector X, an implicit Sigma δ, a binding weight W between the input vectors and the hidden layer, and a set of activation functions that counts the resulting values τ, as figure 13 shows below. The TPM model can be described by three parameters: K (number of hidden nerve cells), N (number of neurons connected to the input per hidden nerve) and L (limit value for weights ({ -L … +L})). [25]

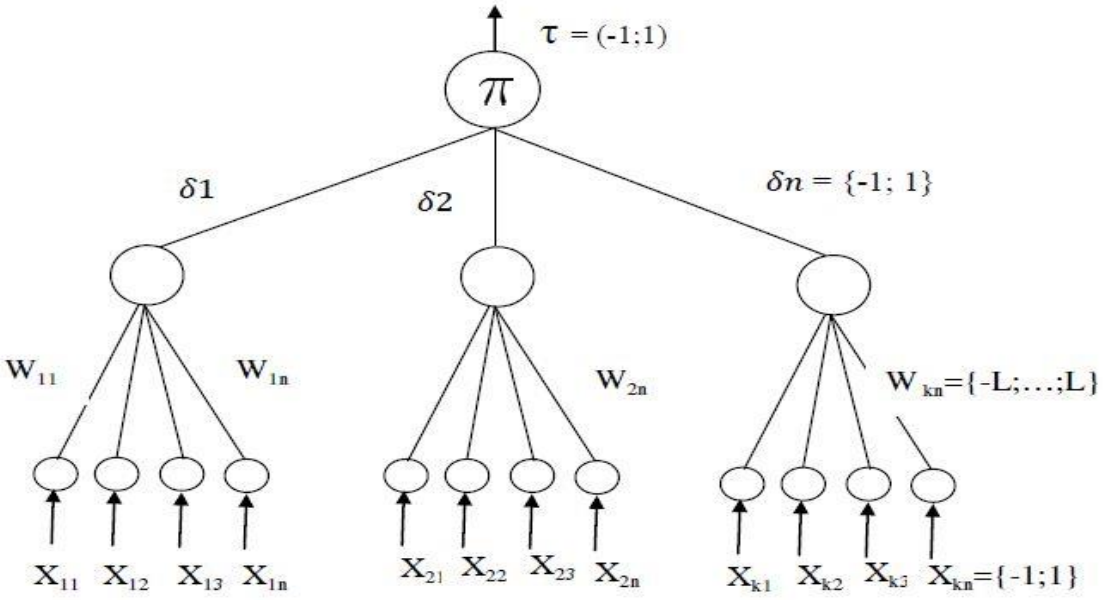

Figure 13 Tree Parity Machine model. Copied from [25]

Two machines have the same neural network structure following a similar TPM pattern. According to Jogdand [25], the output value is calculated by:

$$\tau = \Pi_{i=1}^{k} Sign(\sum_{j=1}^{w} w_{ij} x_{ij})$$

Where:

Hidden neuron: K

Input neuron: $X_{i,j} \in \{-1, 1\}$

Weight value: $W_{i,j} \in \{-, ...0, ...+l\}$

$$Signal(x) = \begin{cases} -1 \; if \; x < 0 \\ 1 \; if \; x \geq 0 \end{cases}$$

The weight value can only be updated if the output value is equal. Here are three different training rules [26]:

$W^{+}_{i,j} = g(W_{i,j} + X_{i,j}.\tau.\theta(\delta i, \tau).\theta(\tau A, \tau B))$ Hebbian learning rule

$W^{+}_{i,j} = g(W_{i,j} - X_{i,j}.\tau.\theta(\delta i, \tau).\theta(\tau A, \tau B))$ Anti-Hebbian learning rule

$W^{+}_{i,j} = g(W_{i,j} + X_{i,j}.\theta(\delta i, \tau).\theta(\tau A, \tau B)$ Random-walk learning rule

Where:

Theta θ is a special function. Theta (a,b) = 0 if $a \neq b$; otherwise, a = b then theta = 1.

g(…) has the function to keep the weights within the range (-L, +L)

x is the input vector and w is the weight vector.

After the two machines are synchronized, their weight matrix is equal. This matrix can be used to construct a shared key between two machines.

The key can be created from the weight matrix as follows:

- On each machine, there is an identical sequence of random number A.
- Size = the length of A / (L * 2 + 1)
- Length of the key = (K * N)/ Size
- Based on the weight matrix, the character position from A is identified to be the key.

After the output values between the two machines are the same, the weight value is updated; therefore, the value of θ(τA ,τB) in the Hebbian training rule is residual. As a result, the following training rule based on the Hebbian training rule is proposed to fit the problem:

$$\boldsymbol{W_{i.j}^{+} = g(W_{i.j} + X_{i.j}.\tau.\theta(\delta i\,,\tau))}$$[27]

Wikipedia contains the information about the attacks on this algorithm. In a public channel, attackers (Eve) can eavesdrop and capture information values between parties A and B, but there is no chance to change them. As a result, attackers cannot hack.

For conventional coding systems, the security of protocols can be improved by increasing the key length. Coding with artificial neurons is improved by increasing the L value of the neural network. This parameter change raises the cost of a successful attack exponentially, which is, therefore, improve the security of key exchange with a high complexity neural network

4.2 Implementation of secret key generation:

The algorithm used for key exchange is presented in the following diagram:

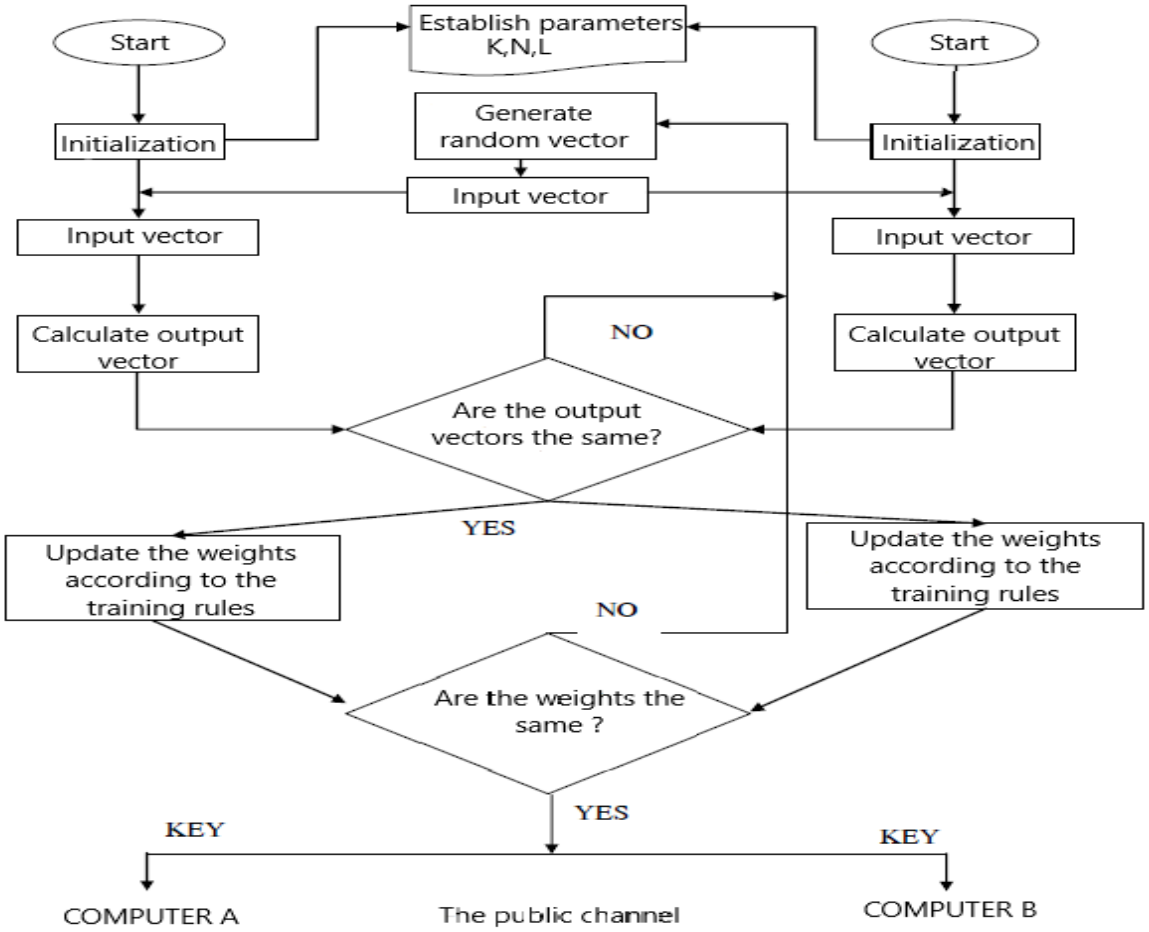

Figure 14 The algorithm of key exchange using Perceptron neural network. Aspired from [26]

There are seven steps to generate a secret key between two machines based on the Perceptron neural network [26], as listed below:

- First, setting up parameters for a neural network is done on both machine A and B
- Secondly, the weights are generated randomly on each machine.
- After that, input vectors are set randomly on machines A and B.
- Then, it is necessary to calculate the output vectors and exchange between machines A and B.
- If the output vectors on both machines are the same means $\tau A = \tau B$

then it is necessary to continue to step 6. Otherwise, step 3 should be repeated.

- The corresponding weights per machine are updated using the Hebbian training rule. If the weights are the same, it is necessary to continue to step 7. If not, step 3 should be repeated.

Finally, the weights of the neural network are the same for both machines. In addition, these weights are used to generate the secret key.

The condition to stop the program is that if the weights of the neural network are the same for both machines. Otherwise, if the weights of the neural network are not identical, it is necessary to repeat step 2.

5. Install the testing program and result

In this report, a neural network model is built for the use in secret key exchange. The C# programming language is used with Visual studio 2010 suite for this program. It is tested on the internal network system model of a company.

The program is demonstrated by following steps:

- Firstly, on the server interface, it is important to choose IP and click on the Start Listener button illustrated in figure 15.

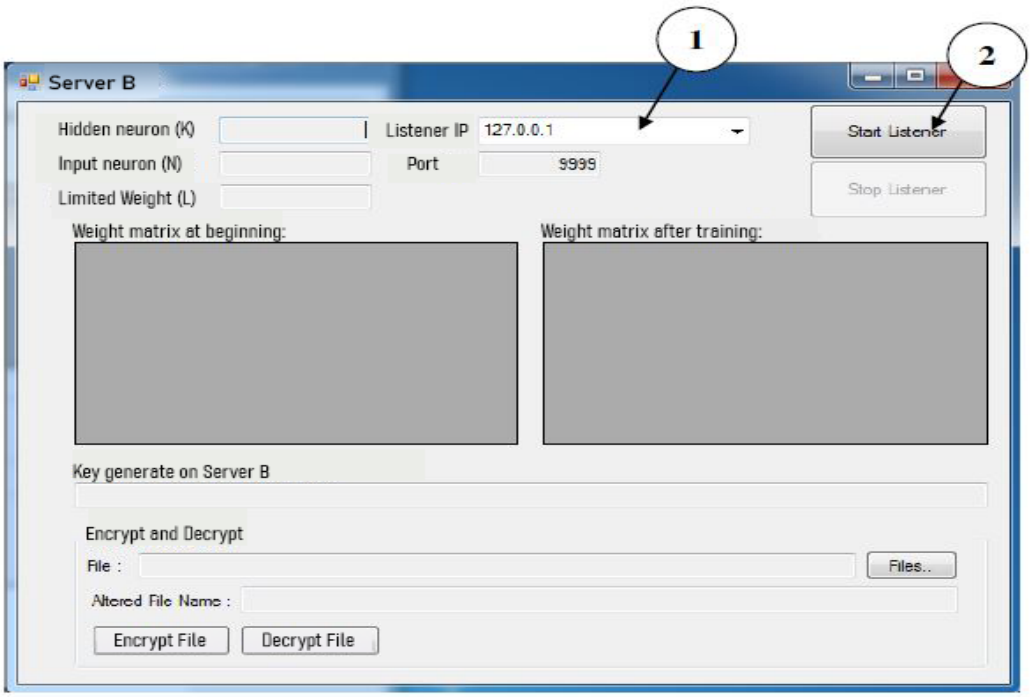

Figure 15 Program interface on server machine

- Secondly, on the client interface, as figure 16 shows, the integer values K, N, and L need to be inputted. Then the server's IP address should be selected, and the Sync with Server button should be pressed.

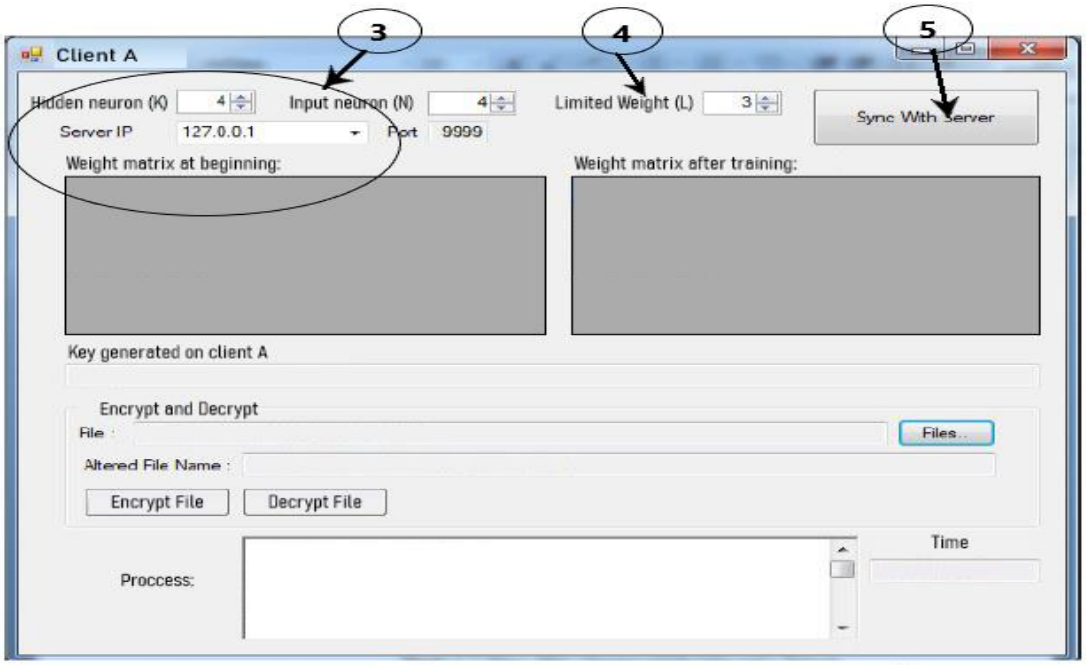

Figure 16 Program interface on client machine

If the weights of the neural network after training are the same for both machines, then the algorithm stops.

If after the setting number of Max (in the problem, $Max = L^4.N.K$) the weight of the neural network is not the same, the program will automatically generate a random new weight matrix on Client and Server, as figures 17 and 18 illustrate.

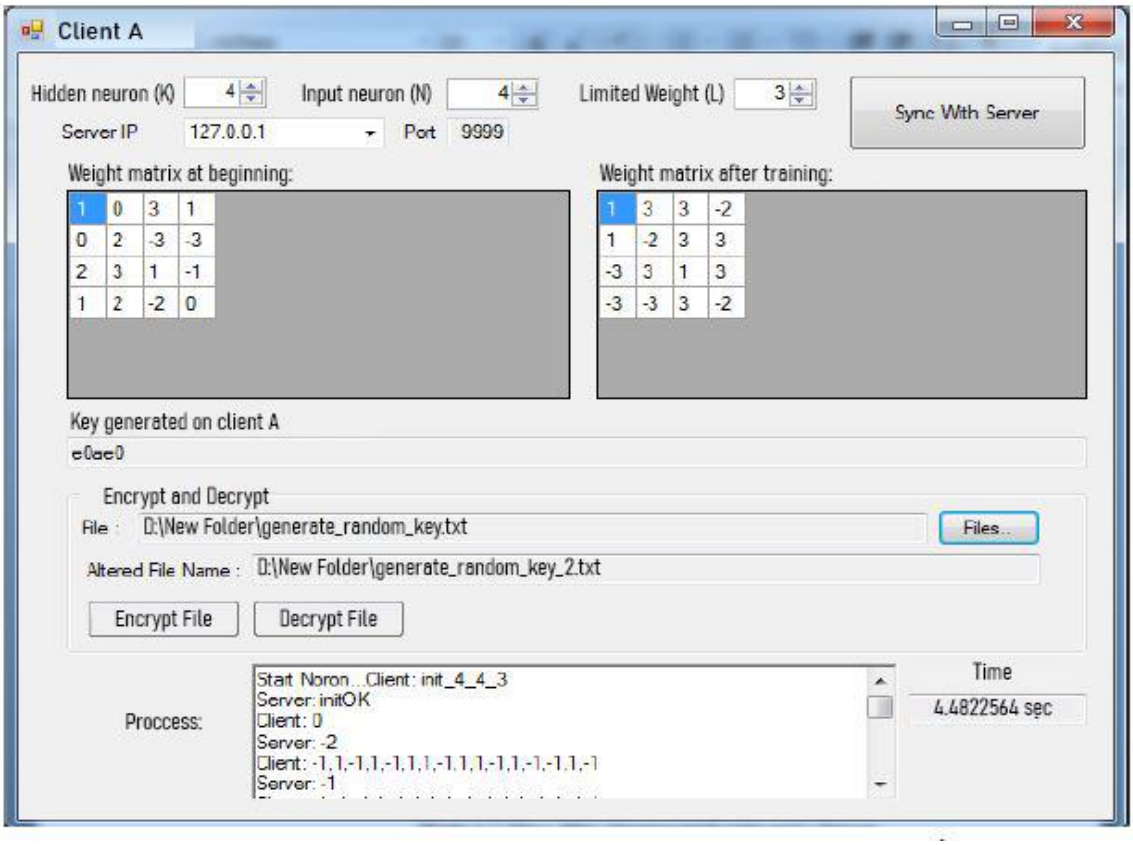

Figure 17 Program interface on client machine after training

Once synchronized, the key is generated based on a training weight matrix. After receive the key, the AES algorithm (128bit) is implemented as the tutorial of Mr.Darcy [28] to encrypt and decrypt files on Server and Client machines.

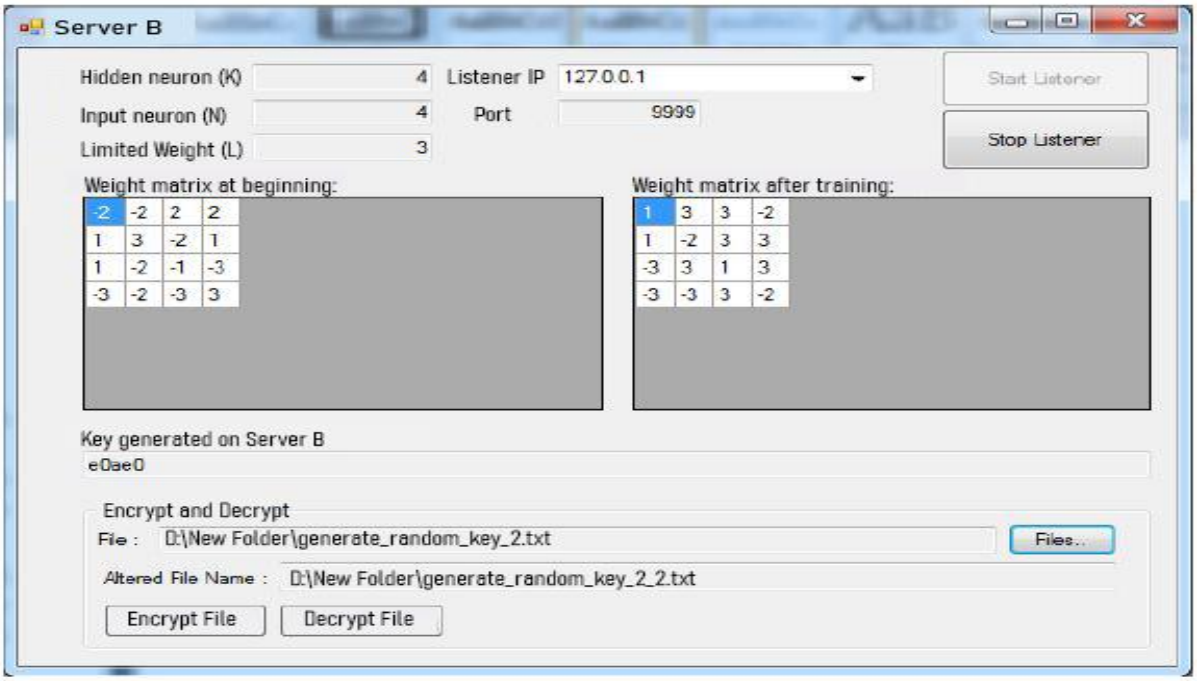

Figure 18 Program interface on server machine after training

As a result, both client A and server B machines produce the same key after the key representing the process of key exchange has successfully been calculated.

6. Conclusion

Through the development of an artificial neural network, it is possible to create a new approach to secure the secret key exchange protocol. The Perceptron neural network key exchange algorithm is a synchronization application, which is suggested to solve the problem.

During the synchronization, each participant only knows their own internal TPM values. As a result, the secret key is generated in the participant's computer without transmitting through an insecure public channel. This process improves the security of the secret key exchange protocol; however, the internal TPM values need to be kept confidential to guarantee the safety of the protocol.

In addition, the program can be changed to be more diversified by updating the hidden layer, increasing or decreasing the number of hidden neurons, changing the value of the weight matrix. This increases the complexity of the algorithm and reduces the attacker's ability to hack the key.

In conclusion, this research shows that the artificial neural network could be used as an alternative for the Diffie-Hellman secret key exchange protocol. The artificial neural network provides a secure method to generate the key which could prevent or raise the cost of the cyberattacks, for instance, man-in-the-middle attacks and brute force attacks.

References

1 Whitfield Diffie and Martin E.Hellman, (1976). “New Directions in Cryptography”, the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21-24

2 Jean-Francois Raymond and Anton Stiglic, (2000). “Security Issues in the Diffie-Hellman Key Agreement Protocol”

3 Menezes A, Van Oorschot P, Vanstone S, (1996). “Handbook of applied cryptography”, pp. 4-10

4 Dr. Ajit Singh, Aarti nandal CSE, SES, BPSMV India, (2013). “Neural Cryptography for Secret Key Exchange and Encryption with AES”, ISSN: 2277 128X, Volume 3, Issue 5, pp. 376-381

5 Preneel B. Understanding cryptography. Berlin: Springer; 2014, pp. 30

6 Stallings W, (1990). Cryptography & Network Security: Principles and Practice. Pearson Australia Pty Ltd.

7 Maryam Ahmed, Baharan Sanjabi, Difo Aldiaz, Amirhossein Rezaei, Habeeb Omotunde, (2012). “Diffie-Hellman and Its Application in Security Protocols”, ISSN: 2319-5967, Volume 1, Issue 2, pp. 69 -73

8 McCulloch, Warren; Walter Pitts, (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. Volume 5, Issue 4, pp. 115–133

9 Neural Networks - History [Internet]. Cs.stanford.edu. 2021 [cited 7 January 2021]. Available from: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

10 McCarthy J, Minsky ML, Rochester N, Shannon CE. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. AIMag [Internet]. 2006Dec.15 [cited 2021Jan.17];27(4):12. Available from: https://ojs.aaai.org/index.php/aimagazine/article/view/1904

11 Minsky M, Papert S, (1969). “Perceptrons”. Cambridge, Mass.-London.

12 P. Werbos, (1974). “Beyond regression: New tools for prediction and analysis in the behavioral sciences,” Ph.D. dissertation, Committee on Appl. Math., Haivard Univ., Cambridge, MA

13 Neural Networks - History [Internet]. Cs.stanford.edu. 2021 [cited 9 January 2021]. Available from: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history2.html

14 Vidushi Sharma, Rachin Rai, Anurag Dev, (2012). “A Comprehensive Study of Artificial Neural Networks”, ISSN: 2277 128X, Volume 2, Issue 10

15 Vidushi Sharma, Sachin Rai, Anurag Dev, (2012). “A Comprehensive Study of Artificial Neural Networks”, International Journal of Advanced Research in Computer Science and Software Engineering 2 (10), pp. 278-284

16 Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu and W. J. Gross (2017), "VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume 25, Issue 10, pp. 2688-2699

17 MA Kramer, (1992). “Autoassociative neural networks”, Computers & Chemical Engineering, ISSN: 0098-1354, Volume 16, Issue 4, pp. 313-328

18 T. Lu, X. Xu, S. Wu, F. T. S. Yu, (1989). ” Hetero associative neural network for pattern recognition”, the IEEE International Conference on Systems, Man and Cybernetics, Cambridge, MA, USA, Volume 2, pp. 658-663

19 Zell, Andreas, (1994), “Simulation Neuronaler Netze [Simulation of Neural Networks]”, German, pp. 73

20 Zamir A, Wu T, Sun L, Shen W, Shi B, Malik J et al. Feedback Networks [Internet]. Feedbacknet.stanford.edu. 2017 [cited 7 December 2020]. Available from: http://feedbacknet.stanford.edu/

21 Freund, Y.; Schapire, R. E. (1999). "Large margin classification using the perceptron algorithm", Machine Learning. Volume 37, Issue 3, pp. 277–296

22 Hastie T, Tibshirani R, Friedman J, Tibshirani R, Friedman J, (2009). Elements of Statistical Learning, Data mining, Inference and Prediction. 2nd ed. New York: Springer.

23 Ojha, Varun Kumar; Abraham, Ajith; Snášel, Václav (1 April 2017). "Metaheuristic design of feedforward neural networks: A review of two decades of research". Engineering Applications of Artificial Intelligence, Volume 60, pp. 97–116

24 Bellman R. A Markovian, (1957). “Decision Process”. Indiana University Mathematics Journal, Volume 6, Issue 4, pp. 679-684.

25 M.Jogdand and Sahana S.Bisalapur, (2011). “Design of an Efficient Neuron Key Distribution Centre”, International Journal of Artificial Intelligence & Applications (IJAIA), Volume 2, No.1, pp. 60-69

26 Ms. Sahana S. Bisalapur, (2011). “Design of an Efficient Neural Key Distribution Centre”

27 Michal Rosen-Zvi, Einat Klein, Ido Kanter, and Wolfgang Kinzel, (2002). “Mutual learning in a tree parity machine and its application to cryptography”, ISSN:1063-651X, 4 August 2002; published 30

28 Mr.Darcy, (2007), http://www.thecodeproject.com/, accessed on 10/01/2020