PASSWORD SECURITY VIA NEURAL NETWORKS

Ass. Lec. Mokhtar Mohammed Hasan
University of Baghdad/ College of Science for Women/ Department of Computer Science

ABSTRACT

Password security and protection are one of the important research topics in modem computer systems. Providing privacy, authenticity, integrity and limited access to data, encryption methods are proposed for password security schemes.

This paper proposes the use of neural network accessing the system, the system needs other information extracted from the user's password along with the password itself, these information is passed to two different neural networks to examine the authenticity of the user, and then decide whether the user is a legal user or an intruder.

The extracted information can be summarized by the time period between each two successive characters in the password and the strength of strike of the user when each character is typed at login time.

As a result is a powerful security scheme for password protection and the user has no wary about password being theft because the related password information can not be theft.

الخلاصـــة

حماية كلمة المرور و امنيتها هي احدى المواضيع المهمة في الانظمة الحاسوبية الحديثة, فمن اجل الحصول على الامنية, الموثوقية, و صحة المعلومات و جعل الوصول الى البيانات الموجودة في النظام يتم فقط من قبل اشخاص مخولين, للاسباب اعلاه نحتاج الى طرق حماية لمفتاح المرور.

الطريقة المقترحة هي باستعمال الشبكات العصبية من اجل الولوج الى النظام, حيث ان النظام المقترح يستعمل معلومات اخرى مستخلصة من كلمة المرور بالاضافة الى قيمة كلمة المرور, حيث ان هذا المعلومات يتم تمريرها على شبكتين عصبيتين مختلفتين من اجل التاكد من موثوقية المستخدم والسماح له بالدخول الى النظام او رفض هذا المستخدم.

المعلومات المستخلصة تمثل الفترة الزمنية بين حرف و اخر عند ادخال كلمة المرور و قوة الضغطة لكل حرف مطبوع اثناء ادخال كلمة المرور, بالتالي هو بناء طريقة جيدة و كفوءة من اجل حماية كلمة المرور, وفي حالة تم اكتشاف كلمة المرور من قبل المتطفلين, فلا حاجة للقلق لان المعلومات المستخلصة من كلمة المرور لا يستطيعون اكتشافها

KEY WORD :
Security, Protection, Back Propagation Neural Network, Learning, Testing

INTRODUCTION

The mechanisms for security and protection of a computer system can be classified into three concentric circles ([8]):

- The innermost circle represents the memory of a computer—RAM and disk mechanisms such as base-bound registers, virtual memory mappings, and file access concern Trojan horses, processes entering supervisor state and gaining

supervisor privilege, and processes attaining super user privilege.

- The middle circle represents the security perimeter of a system: only authorized people are allowed to cross the perimeter and establish processes within; their processes are controlled by the mechanisms of the inner circle.

- The outer circle represents the network—all the other computers and people who want to interact with a given one. Here the concern becomes the ability to complete exchange transactions successfully (the central notion of commerce and collaboration). The biggest problem is authentication. Many of the vulnerabilities of networked systems arise from inadequate means authenticate users and machines. Sophisticated cryptographic protocols have been devised to assist with such aspects as secret communication, digital signatures, certificates, and money.

This paper reviews password security and protection, in a computer system and focuses on password protection using neural networks. Especially, a new method of password protection is invented.

There are many ways of password protection used in the current systems scaled from the easy methods such as encryption process of password for small and minor importance system to creating accounts for each user with a system list of password with some constraints for choosing the password itself using the access control list and access matrix found in the system that has a high importance.

THE PROPOSED SYSTEM

In order to prevent the password from being attacked by the pentrators, we have to design a good password features, these features represent the combinations of the characters in the password itself as well as the information that are taken from the password typing. The proposed system uses three different information for password protection:

- The value of the password itself
-The time period between each two successive password's characters (may be letters, or special characters and so on).
- The key strike of the characters

If we notice that, the password characteristics are difficult to achieve by the pentrators which are the strength of the key strike and the time period between each two successive characters, even if the pentratos can guess the password characters in some how, he can not for sure obtain the other password's characteristics which are the time period between each two successive characters and the strength of the key strike, because these information is supplied to the system at the login time by only the actual user who created the password and information is not saved in a system password file because we use neural networks for analyzing the correctness of such information, so, the password protection is achieved.

NEURAL NETWORKS FOR PASSWORD PROTECTION

The aim of a neural network is to recognize the other information extracted from the password at login time, let us rewrite these information again:

- The time period between each two successive characters in the password (PT as a shortcut for time period between password characters)
- The strength of the strike for each character in the password (ST as a shortcut for strike of characters in password)

As I mentioned, the extracted information is not saved in any system file to prevent accessing by other intruders, so the neural network will be used to recognize PT and ST information in order to check the user authenticity a long with the password itself.

We used a neural network called Back Propagation Neural Network that is developed by Paul Werbos [1], this net used here for the following purposes:

- Very popular model in neural networks
- Easy to training Can estimate the behave of the input patterns
- Supervised training algorithm

The estimation property means that, the user ,for example, have one character password length, and PT =300 milliseconds, the user want to login the system, he can not repeat the same operation with PT=300 milliseconds every time he logs in the system, so the neural network manages this input with acceptable range of error.

The supervised property of this algorithm made it suitable for the proposed system because I have the input features represented by the PT and ST after formulating it to a suitable input codes, and these information should indicate exactly one user after matching with the value of the password itself, in other words, the system have the input features and the desired output user which is the supervised version of neural network is suitable for this purpose.

AN OVERVIEW FOR A BACK PROPAGATION NEURAL NETWORK

There are two stages in this network:

- The learning stage of the neural network

There are two phases in its learning cycle, one to propagate the input pattern and the other to adapt the output. It is the error signals those are backpropagated in the network operation to the hidden layer (s). It does not have feedback connection, but errors are backpropagated-during training, least mean squared error is used.

The input patterns represent by the PT and ST information as mentioned before, and the output represents the authenticity measure of that user.

This neural network composed of three different layers, the input layer which take the input pattern, and the hidden layer, and the output layer which represents the neural network response, a further point is that the network shown in Figure (5) is fully connected, which means that the output of every neuron in one layer is connected to an input of every neuron in the next layer, starting from the input layer and ending at the output layer. Not every multi-layered perception is connected in this way but this is the most common way of doing it.

Errors in the output determine measures of hidden layer output errors, which are used as a basis for adjustment of connection weights between the input and hidden layers. Adjusting the two sets of weights between the pairs of layers and recalculating the outputs is an iterative process that is carried on until the errors fall below a tolerance level. Learning rate parameters scale the adjustments to weights. A momentum parameter can also be used in scaling the adjustments from a previous iteration and adding to the adjustments in the current iteration.

- The testing stage of the neural network

Once training is completed, the weights are set and the network can be used to find output for new inputs. The dimension of inputs are limited by the number of neurons in the input layer, and the dimension of outputs are limited by the number of neurons in the output layer.

- Applying Neural Network in the System

The proposed system uses two different neural networks:

1) PTNN (Period Time Neural Network) : that is used for PT features
2) STNN (Strike Time Neural Network) : that is used for ST features

Both neural networks have the same general structure with same input, hidden and output neurons, the difference is with the type of the input information as an input pattern for each network, after the user start entering the password, the system collect the PT information and ST information, and when user finishes entering the password, the system search in system list of password in order to recognize the user as a first step, when matching is found, the system takes the user number that is assigned for each user inside the system, like user 1 has number =1, user 2 has number =2, and so on, this number taken and combined with PT and ST information to form two input vectors, PTNN input vector, STNN input vector, and use the corresponding neural network to recognize that vector pattern, if both of neural network responds for that user, the user will be allowed to enter the system, if one of them fails or both fails , the access is denied for that user.

PTNN

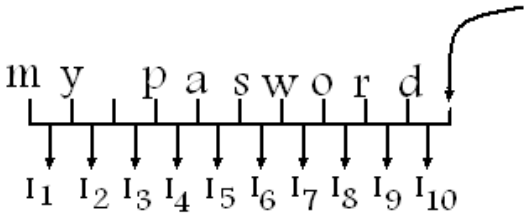
This neural network is used to recognize the PT feature of the current user, this network has 111 input nodes, 223 hidden nodes, and one output node, the input information here represented by the PT features as well as the user number to form the input vector for that neural network.

CALCULATING INPUT NEURONS

The longest password length in this system is 10 characters, and if we compute the numbers of pairs in this longest password we find 9 pairs with 9 elapsed times, and the time elapsed between the finish typing the last character in the password and pressing the login key, so we have 10 periods each of which represented by 10_bits, the time measure here in milliseconds by doubling the number; 50 milliseconds wait means 100 milliseconds; 500 milliseconds wait means 1000 milliseconds (1 second), so we have a maximum time of 2^{10} *2=2048 milliseconds=2.048 seconds elapsed time between each two successive characters the user can range, so we have 10 different input information each of 10_bits = 100_bits=100 neuron plus 10_bits for user number=110_bits=110 neurons, plus one for bias the total is 111 nodes.

Assume that the password was "my pasword", 10 characters, so there are 9 pairs of two successive characters in this password, and the tenth pair is between writing the last password character and pressing the login key:

Transferring and Pressing to Login Button



Where I=Information, so, each information takes:
*10_bits = 10_information * 10_bits_for_each=100_bits*
We need extra 10_bits for user number, so, the total of:

100_bits + 10_bits_for_user_number = 110_bits = 110 neurons
110 + 1 for bias = 111 neurons at the input layer

CALCULATING HIDDEN NEURONS

The number of hidden neurons are estimated, there is no ideal equation to compute the number of hidden neurons in back propagation neural network, so this number is depend on the experimental results and testing, some references emphasis on the following equation that we use it:

$$Hidden\_neurons = Input\_neurons * 2 + 1$$

Table (1) shows the different number of hidden neurons compared to the number of cycles need to learn the neural network assuming the system consists of 20 users, so we get the final number of hidden neurons.

Table (1) _ preliminary experiment results

Hidden number	Iterations number	Error
20	4000	0.3544
50	4000	0.3541
130	4000	0.0219
160	4000	0.0195
190	4000	0.0183
223	4000	0.0031

CALCULATING OUTPUT NEURONS

The number of output neurons used in this system are just one, after supplying the input vector features for the neural network, the output will be an authenticity measures that scaled between 0 and 1, as the output value goes to one, the user is more authenticate, as the output value goes to 0 the authenticity reduced, so we need a threshold to recognize between authentic users and no authentic users, this threshold decided to be 0.7, some errors are allowed here because it is difficult to the user to

supply the same exact PT and ST in each time which is one of the reasons of using back propagation neural network, figure (5) shows the layout of PTNN.

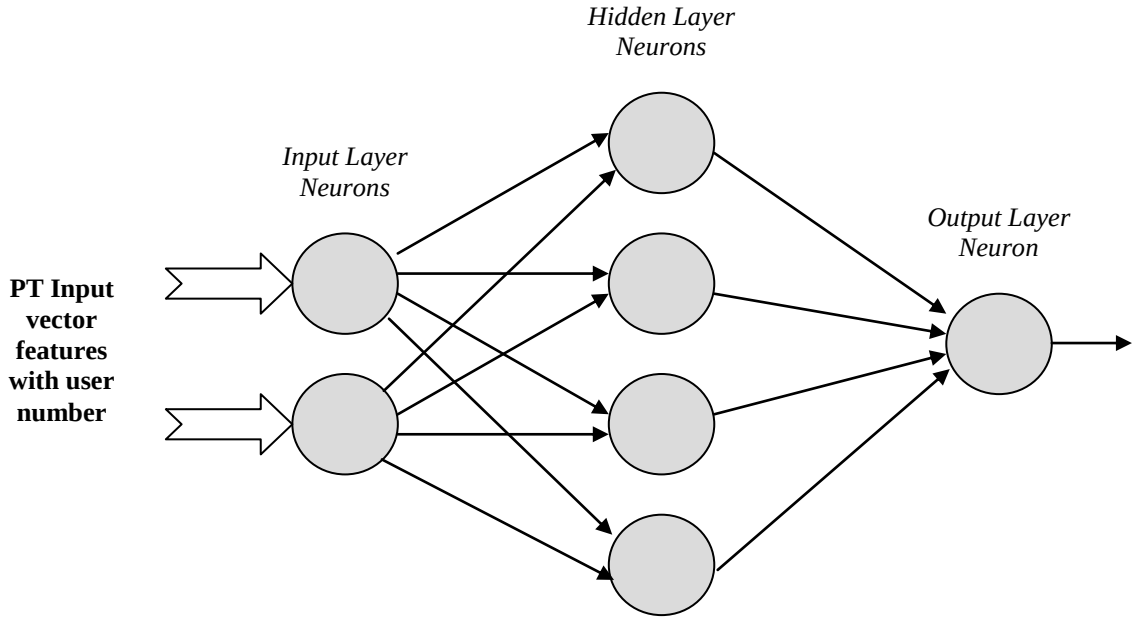


Fig (5) _ the layout of PTNN

STNN

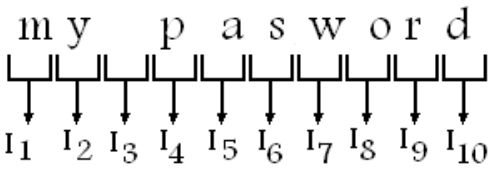
The design of this neural network is identical to this one that used in PTNN except that the codes in the input vector represents different kind of information, this network also has 111 input nodes, 223 hidden nodes, and one output node, the input information here represents the ST features a long with the user number to form the input vector for that neural network.

CALCULATING INPUT NEURONS

The longest password length in this system is 10 characters, each character has its own ST, so the system computes 10 different periods for ST, also each character has 10_bits representation which allows 2.048 seconds the ST of each character in the password.

The ST for each character is computed by computing the time elapsed between holding down the keyboard key and releasing up this key which represents the ST for that character, these 10 different periods as well as the user number are presented to the neural network in order to decide the authenticity of that user.

As an illustration example, assume that the password was "my pasword", 10 characters; the ST information will be computed from:



So, each information takes 10_bits = 10_information * 10_bits_for_each=100_bits
We need extra 10_bits for user number, so, the total of:

100_bits + 10_bits_for_user_number = 110_bits = 110 neurons in input layer
110 + 1 for bias = 111 neurons

Calculating Hidden Neurons

The number of hidden neurons are 221 in the same way computed in section (5.2.1.2).

CALCULATING OUTPUT NEURONS

There is one neuron used as an output neuron as illustrated in section (5.2.1.3), figure (6) shows the layout of the STNN**.**

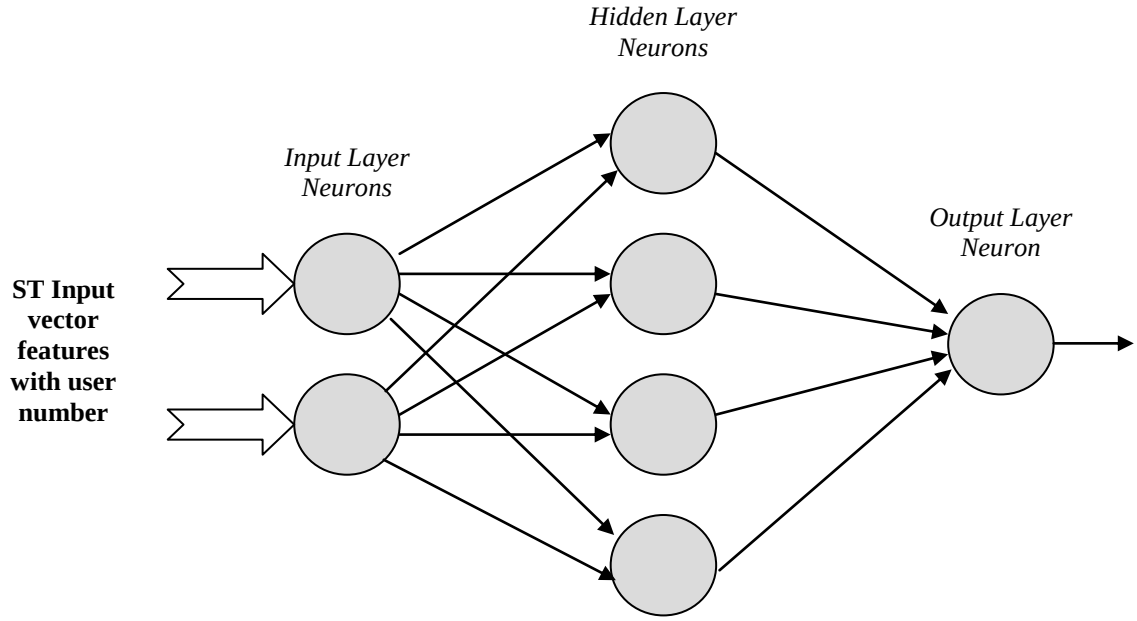


Fig (6) _ the layout of STNN

LEARNING STAGE OF THE PTNN AND STNN

Once the required system is installed on a target computer, the system needs now to learn the neural network to the available users in this system so the weights are saved.

Any new registered user in the system, his ID and Password will be taken as tradition, the system store the ID, password, user number that is generated by the system, and the extracted information from the password at the moment of typing the password will be used to learn the neural network, as a result, we can use the neural network at log in for registered users.

The Back-propagation network undergoes supervised training with a finite number of pattern pairs consisting of an input pattern and a desired or targets output pattern. An input pattern is presented at the input layer. The neurons here pass the pattern digits to the next layer neurons, which are in a hidden layer. The outputs of the hidden layer

neurons are obtained by using perhaps a bias, and also a threshold function with the activation determined by the weights and the inputs. These hidden layer outputs become inputs to the output neurons, which using possibly a bias and a threshold function with their activation to determine the final output from the network.

Learning Algorithm:

- Let x0(0), x0(l), x0(2), up to x0(111) be the input vector features to the input layer.
- Let y1(0), y1(1), y1(2) up to y1(223) be the input of the hidden layer, which is the output of the input layer.
- Let y2 be the final output value.
- w1(i ,j) is the weights connection between input node i and hidden node j
- w2(i) is the weights connection between hidden node i and output node
- x1(1), x1(2), up to x1(111) and x2 is a temporary storage

As a first step, the system has to initialize all weights with a random number scaled between 0 and 1 an d must be not equal to zero.

The system has to compute the hidden neurons which is:

$$x1(j) = \sum_{i=0}^{111} w1(i, j) * x0(i) \quad \textbf{… Equation 1}$$

We need a derivative function to ensure that the output will fall in the range 0 to 1; the one that is most often used successfully in multilayered perceptrons is the sigmoid function, shown in Figure (7).

So, using the sigmoid function, the hidden layer inputs can be computed from the following equation:

$$y1(i) = \frac{1}{(1 + e^{-x1(i)})} \qquad \textbf{… Equation 2}$$

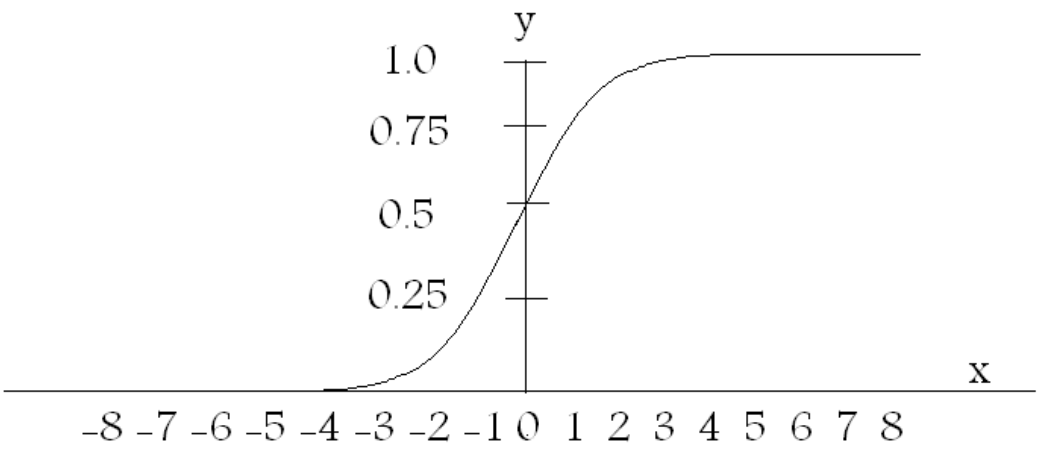


Fig (7) _ sigmoid function

For positive values of x, as x increases y approaches 1. Similarly, for negative values of x, as the magnitude of x increases y approaches 0. In addition, when x = 0, y = 0.5, So the output is continuous between 0 and 1 and is therefore differentiable.

The temporary output of the hidden layer is computed from the following equation:

$$x2 = \sum_{i=1}^{223} w2(i) * y1(i) \qquad \textbf{… Equation 3}$$

Using the sigmoid function, the final output can be computed from the following equation:

$$y2 = \frac{1}{(1+e^{-x2})} \qquad \textbf{… Equation 4}$$

If the actual Neural Network output which is (y2) is far a way from the desired output, we have to go to the next step, which is Weight Adaptation..

Use a recursive algorithm starting at the output nodes and working back to the first hidden layer Adjust weight by:

$$w2_i(t+1) = w2_i(t) + \eta * \delta * y1_i \qquad \textbf{… Equation 5}$$

Where

η : is a gain term

δ : is an error term for output node which can be computed from the following

equation:

$$\delta = y2*(1-y2)*(d-y2) \quad \text{... Equation 6}$$

And similarly, d is the desired output of the output node which is 1.

The weight adaptation between input and hidden layer, Equation 5 is used by updating w1 using y2, the equation of δ will be:

$$\delta_j = y1*(1-y1)*\sum_k \delta_k * w_{jk} \quad \text{... Equation 7}$$

Where k is the overall nodes in the layer above node j, internal node threshold are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant valued inputs.

Convergence is sometimes faster if a momentum term is added and weight changes are smoothed by:

$$w1_{ji}(t+1) = w1_{ij}(t) + \eta * \delta_j * x0_i + \alpha * w1_{ij}(t) - w1_{ij}(t-1) \quad \text{... Equation 8}$$

Where $0 < \alpha < 1$

Note that the weights are updated after each pattern is presented and not after the whole training set is presented. The reason why this is done is because the training set is probably very large, so that the time taken to train becomes intolerable. This has not been shown to be equivalent to minimizing the mean squared error, but is widely adopted.

The above algorithm is repeated until the neural network is learned. Figure (8) shows the flowchart of the learning algorithm.

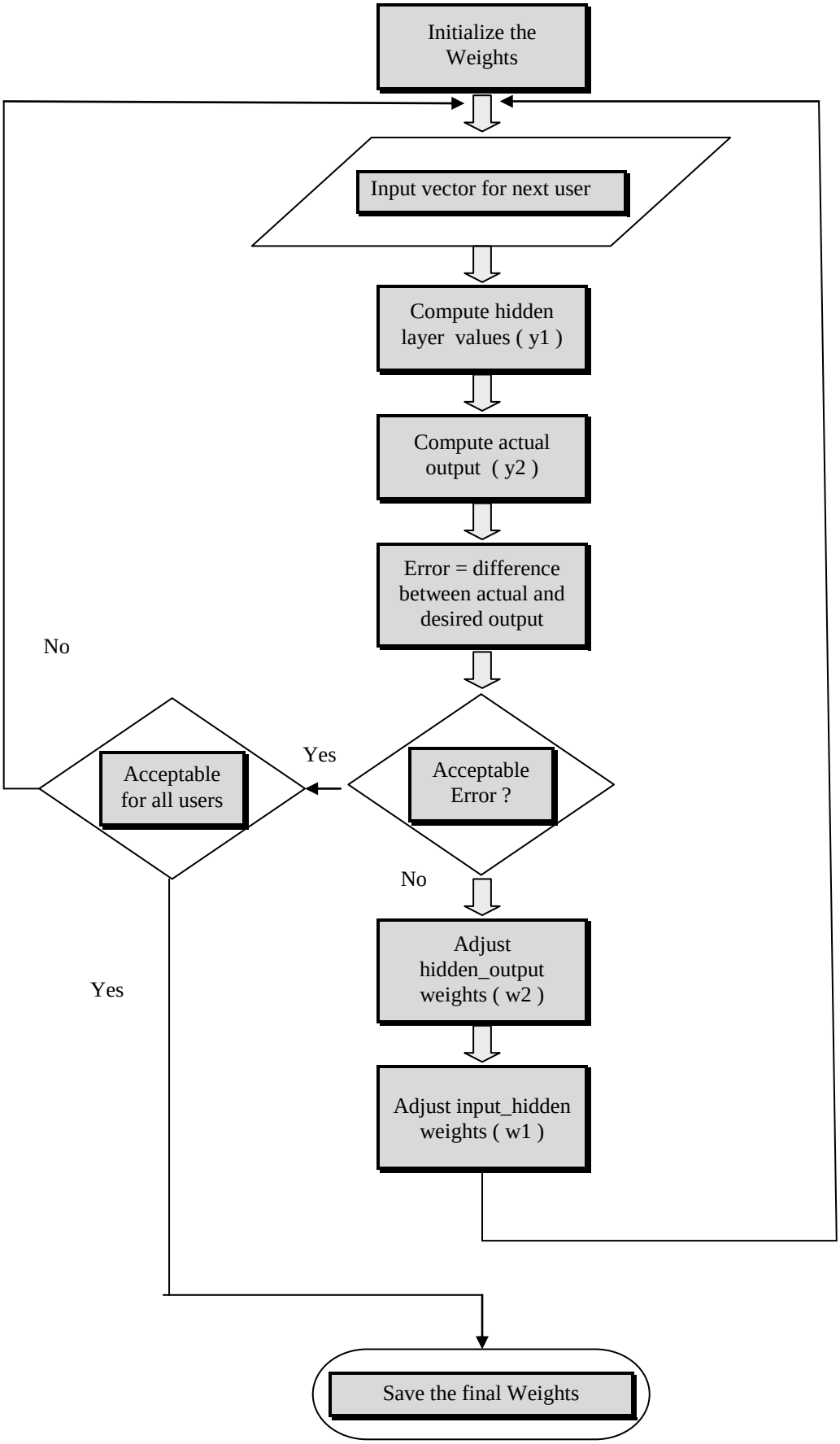


Fig (8) _ the learning process

Testing Algorithm:

- Let x0(0), x0(l), x0(2), up to x0(111) be the input vector features to the input layer.
- Let y1(0), y1(1), y1(2) up to y1(223) be the input of the hidden layer, which is the output of the input layer.
- Let y2 be the final output value.
- w1(i ,j) is the weights connection between input node i and hidden node j
- w2(i) is the weights connection between hidden node i and output node

The system has to compute the hidden neurons which is:

$$x1(j) = \sum_{i=0}^{111} w1(i, j) * x0(i) \quad \textbf{… Equation 1}$$

Using the sigmoid function, the hidden layer inputs can be computed from the following equation:

$$y1(i) = \frac{1}{(1 + e^{-x1(i)})} \quad \textbf{… Equation 2}$$

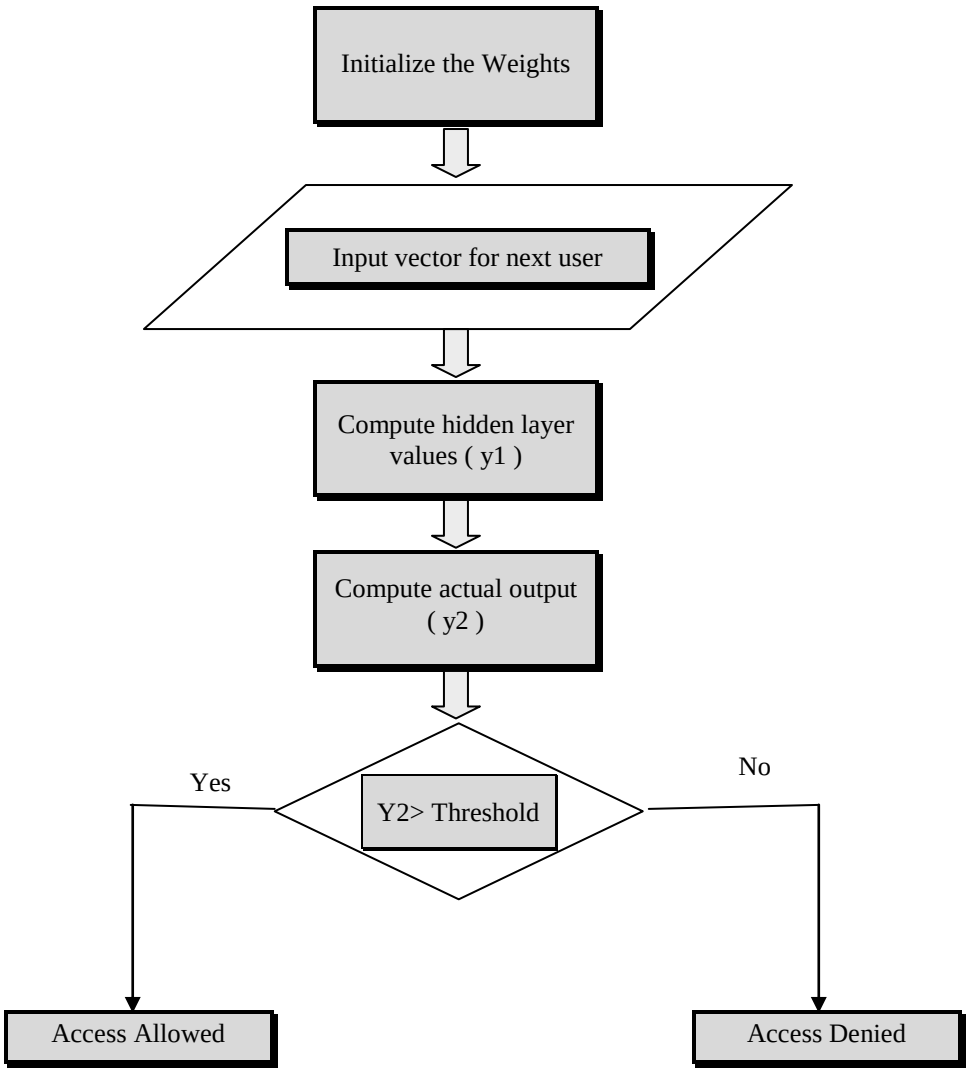
The temporary output of the hidden layer is computed from the following equation:

$$x2 = \sum_{i=1}^{223} w2(i, j) * y1(i) \quad \textbf{… Equation 3}$$

Using the sigmoid function, the final output can be computed from the following equation:

$$y2 = \frac{1}{(1 + e^{-x2})} \quad \textbf{… Equation 4}$$

If the output (y2) larger than the threshold value, then the user is allowed to access to the system, otherwise the user is denied access to the system. Figure (9) shows the flowchart of the testing algorithm.



Flowchart (9) _ testing process

EXPERIMENTAL RESULTS

I applied two examples on the suggested system, the first example is acceptable one, and the second example is rejected by the system, let us consider the user number 12 in the system, when the system create an account for that user and the user enters the user name and password for the first time, the system extracts these information from the password, these information after multiplying the PT and ST information by 2 as mentioned before is:

Password Value	
collection	
PT information	ST information
790, 680, 704, 786, 696, 652, 804, 704, 644, 978	***304, 378, 254, 518, 420, 382, 388, 512, 418, 420***

So that, the system will store the weights of the neural network after learning using the above information which will be used at login time.

ACCEPTABLE EXAMPLE

Assume that a user has the following information registered in the system:

User Name : beginner

Password : collection

The user number in the system was 12, so, at the login interface the user will type his user name and when switches to the password entering area and starts typing the first letter of the password which is letter 'c', the system starts to record two type of information which is PT and ST information, as mentioned before, according to the speed of that user that he decided using my proposed system, we get the following time periods in milliseconds and already multiplied by 2.

PT information	ST information
706,748, 766, 680, 758, 698, 736,682,664, 980	***442, 464, 398, 424, 396, 462, 448, 388, 368, 378***

User Number

12

The user number will be used for both PT and ST information.

The PTNN input vector features will be extracted from the above information as well as the STNN input vector features, Figure (10) and figure (11) shows the final PT and ST vectors that will be used as an input to both PTNN and STNN respectively.

PTNN input

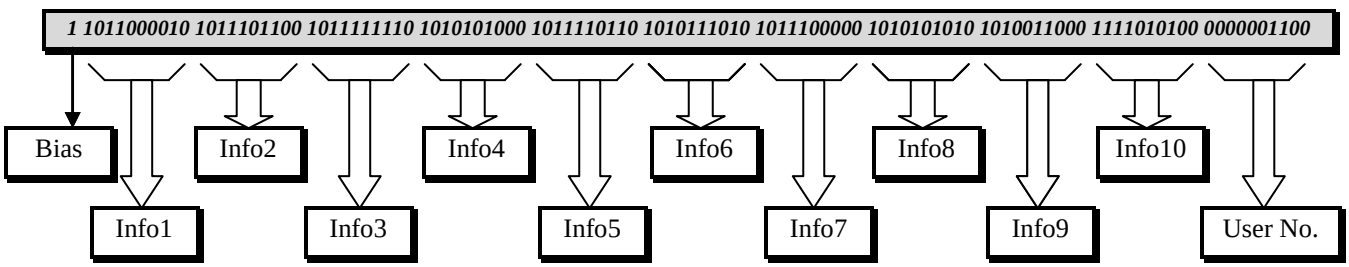


Fig (10) _ the input vector information for PTNN

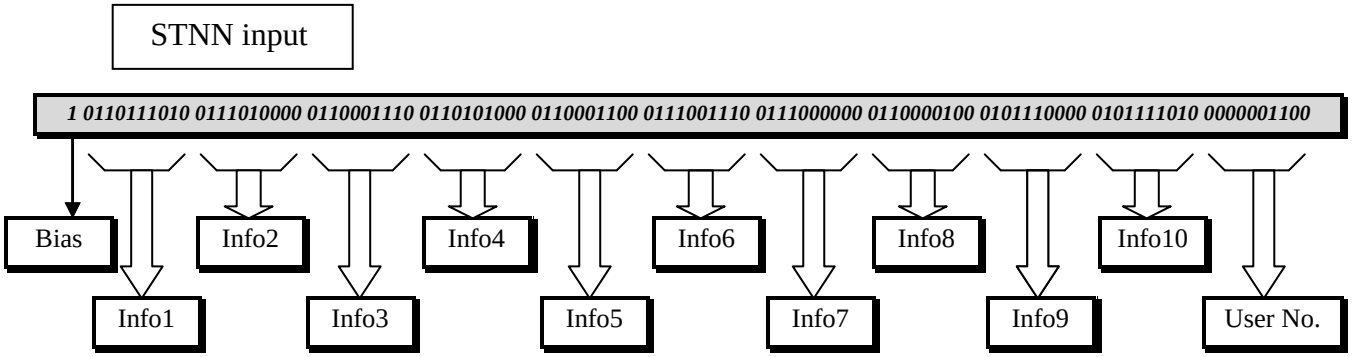


Fig (11) _ the input vector information for STNN

By applying each input vector to the corresponding neural network, i.e. by giving the PT vector features to the PTNN and ST vector feature to the STNN, we will obtain two different outputs each one from a network, these two outputs are:

PTNN output : 0.897 which is > 0.7

STNN output : 0.799 which is > 0.7

As noted above, the user is authenticated user, so this user is allowed to enter to the system.

REJECTED EXAMPLE

I applied the system on the same previous user:

User Name : beginner

Password : collection

The user number in the system was 12, so, the information that extracted from the user password at typing time is:

PT information

918,988, 834, 922, 812, 942, 898,864,890,968

ST information

788, 846, 738, 372, 358, 298, 324, 896, 842, 880

User Number

12

As you noticed, the PT and ST information is quiet different from the original user information, that is mean, the user is not authenticated that steals the password and tries to log into the system, let him do what he came for.

The PTNN input vector features will be extracted from the above information as well as the STNN input vector features. Figure (12) and figure (13) shows the final PT and ST vectors that will be used as an input to both PTNN and STNN respectively

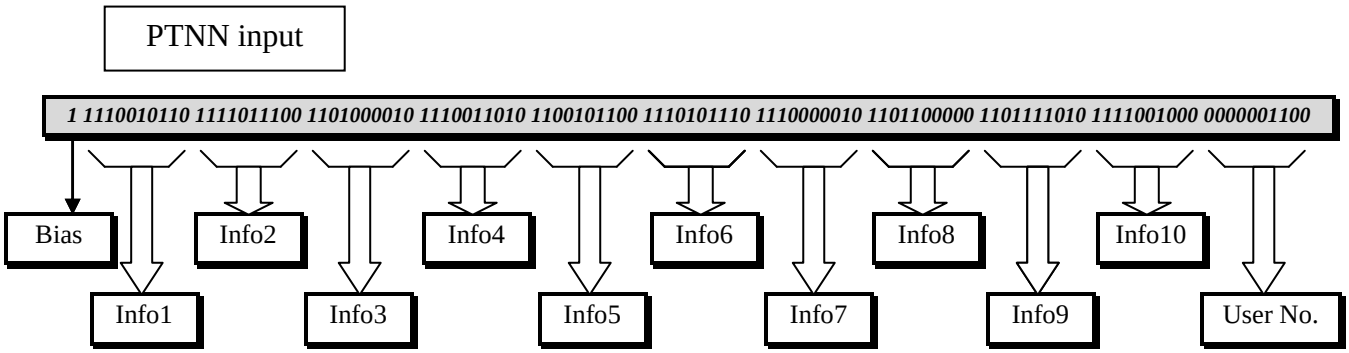


Fig (12) _ the input vector information for PTNN

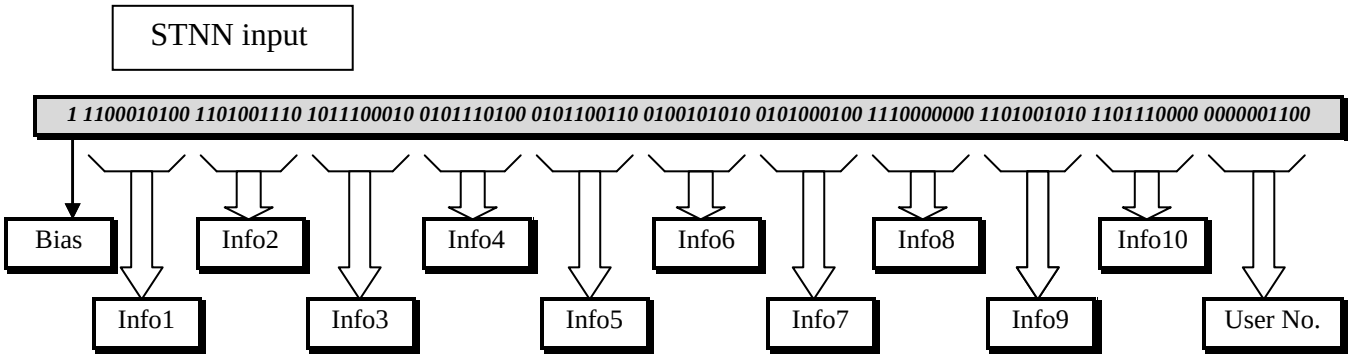


Fig (13) _ the input vector information for STNN

By applying each input vector to the corresponding neural network, i.e. by giving the PT vector features to the PTNN and ST vector feature to the STNN, we will obtain two different outputs each one from a network, these two outputs are:

PTNN output : 0.531 which is <= 0.7

STNN output : 0.242 which is <= 0.7

As noted above, the user is not authenticated user, so this user can not login into the system, which is the aim of this paper.

CONCLUSIONS

In this system, I built a strong enough security method for password because all the newer systems use the password as the live key for their users and the system is aware more than the user for the password security and provision new techniques for protection, this is a powerful method for protection, that uses the password value as well as the information extracted from the password itself which called PT and ST, if the pentrators can guess the password in some how, he can not guess the PT and ST information and I can say that it is impossible to guess these information, so, when the password is protected, the system is protected.

By using the Neural Network for password security, it has the main significant property which is the acceptable of input within a specific range of error, because the extracted information from the user's password at login time is not the same in each time the user logs in to the system and can not be so, there is a different in PT and ST because the time measured in milliseconds, so the neural network fits our demand in this case and it accepts the extracted user information at login time with a suitable error.

REFERENCES

- Werner Kinnebrock, “Neural Networks, Fundamentals, Applications, Examples”, Technical University Rheinland-Pfalz, 2nd Revised Edition, 1995
- Kung S. Y., “Digital Neural Network”, New Jersey, 1993
- Simon Haykin, “Neural Networks, A comprehensive Foundation”, McMaster University, Hamilton, Ontario, Canada, 1994
- Richard P. Lippman, "An Introduction to Computing with Neural Nets", IEEE ASSP magazine, 1987
- Jacek M. Zurada, “Introduction to Artificial Neural Systems”, 1996
- John R. Smith, “Feature Extraction for Neural Network”, March 6, 1996, Internet Paper
- Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer and Terence C. Fogarty, "Lecture Notes in Computer Security", First European Workshop, Euro 99, Paris, France, April 1998.
- Henk C. A. Van Tilborg, "An Introduction to Cryptology", Kluwer Academic Publishers, 1988
- B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code", Second Edition, John Wiley & Sons Inc, 1997.
- Alan G. Konheim, "Cryptography: A Primer", John & Wily, 1992.