29

Designing and Training Feed-Forward Artificial Neural Networks For Secure Access Authorization

Fadi N. Sibai, Aaesha Shehhi, Sheikha Shehhi,
Buthaina Shehhi and Najlaa Salami
UAE University
United Arab Emirates

1. Introduction

Password-based authorization is a key security feature to gain access to accounts, files, and like PINs, can be used to control access to secure rooms, cabinets, electronic equipments, control panels, and other valuables. It relieves employees from carrying physical keys or smart cards and does not require the use of more expensive biometric devices. Memorizing a username or user number of ID and a matching password is sufficient to get access to the controlled computer account, file, building area or equipment. Typically, a key pad is required to key in the user ID followed by the password, non-volatile storage to record and save the matching (user ID, password) combination, and some controller for controlling the password detection operation. The user ID and passwords entered by the user via the key pad are latched (L) and the memorized (user ID, password) pairs on record are retrieved from storage, and then compared with the (user ID, password) combination entered by the user via the key pad. If there is a match the user is given access, otherwise the reading of saved (user ID, password) combinations on record in the storage are read one by one until a match is found, or the list of pairs are exhausted in the storage and the user is denied access. Other alternative implementations are possible. The major weakness of such digital system implementations is the possibility to steal saved (user ID, password) pairs by probing the bus connecting the storage to the data path containing the comparator as depicted in Fig. 1. Such insecurity can be avoided by encrypting the (user ID, password) pairs prior to saving them in storage, and then decrypting them after reading them from storage and prior to performing the comparison operation. However this requires either more programmable and fast hardware which executes encryption and decryption in software or a dedicated encrypt/decrypt hardware engine. If the encryption scheme's keys are compromised, then this probing technique can still reveal the matching (user ID, password) combinations. Another security vulnerability of storing the user ID and passwords combinations in a table saved in storage is that even though the user IDs and passwords may be encrypted, the hacker may modify table entries or create new table entries with fake user ID and password combinations.

An alternative is to employ an artificial neural network circuit which "memorizes" the matching (user ID, password) combinations and generates the matching signal as a result of presenting the user-entered (user ID, password) pair via the key pad. Such an artificial neural network achieves this operation by being trained with a set of (user ID, password, output) triplets where the output is 1 when the password matches the user ID, and 0 otherwise. In the training phase, the artificial neural network refines its link weights to more closely match the outputs of each training line. In the training set, the majority of the triplets will have an output of 0 (non-matching) and only the matching combination of user ID and password will have an output of 1. Such an artificial neural network has the advantage that it can be implemented on a single chip and does not require the matching (user ID, password) combinations to be stored on external storage, but requires that the network weights be stored. While these can be stored on-chip making the information stealing attempt very difficult, the weights can be still be stored externally and retrieved from external storage. If the weights are stolen while being read from external storage by probing, this stolen information is of little value as it makes the process of deriving the matching (user ID, password) pair very difficult. Indeed, the matching (user ID, password) pair is "memorized" and integrated in to the neural network's weights and without knowing the internal organization of the network (e.g. whether feedforward, presence of feedback loops, number of network layers and number of neurons in each layer) and the exact order of the weights and which neural link they correspond to, the process of extracting the matching (user ID, password) pair is virtually exhaustive.

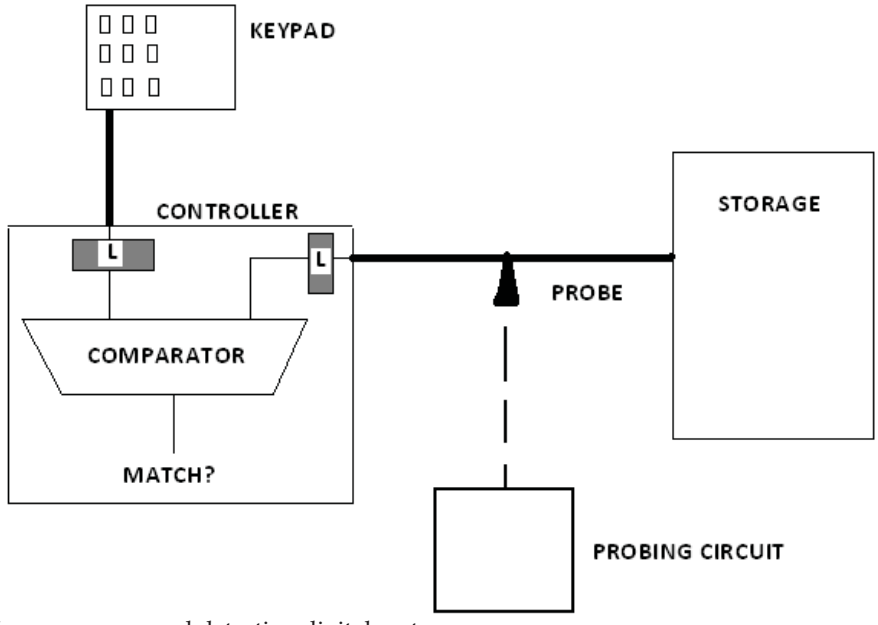


Fig. 1. Insecure password detection digital system

In this Chapter, we present our research work on feed-forward artificial neural networks for secure access authorization. In Section 2, we briefly review artificial neural networks. In

Sections 3 and 4, we review previous work on this subject and present our artificial neural network organization, respectively. In Section 5, we present our data coding for binary and analog inputs. Our network's training, simulation and testing is then described in Section 6. The accuracy results of all the variation of trained and tested networks are then presented in Section 7. Section 8 concludes the Chapter.

2. Artificial Neural Networks

Artificial neural networks model biological neural networks in the brain and have proven their effectiveness in a number of applications such as classification and categorization, prediction, pattern recognition and control. An artificial neural network consists of an interconnected group of artificial neurons. Such a network performs computation and manipulates information based on the connectionist approach in a similar but simpler fashion than the brain would perform. Many types of artificial neural networks (Faussett, 1994) exist including feed-forward neural networks, radial basis function (RBF) networks, Kohonen self-organizing networks, recurrent networks, stochastic neural networks, modular neural networks, dynamic neural networks, cascading neural networks, and neuro-fuzzy networks. Multi-Layer Perceptron (Rumelhart & Mclellan, 1986; Haykin, 1998) (MLP) are perhaps the most popular, where neurons in a feed-forward type network perform a biased weighted averaging of their inputs and this sum is then subjected to a transfer function, in order to limit the output value.

As depicted in Fig. 2, a neuron is made of a cell body bordered by a membrane, inside of which is a nucleus, across which incoming electric (or chemical) signals composed of polarised ions arrive via neuron inputs known as dendrites. The neuron output or outgoing signal travels over a connector or terminal known as axon --where the neurotransmitters reside – and which connect to other neuron dendrites via synapses. Ten thousand of neuron types are known to exist in the brain of different shapes and terminal densities. A neuron may have thousands of dendrites but only one axon. A neuron output –axon—connects to another neuron input –dendrite—in an area called a synapse where the axons terminate. When enough positive ions gather inside the neuron's membrane the neuron fires, i.e. a large electric signal is generated and travels out over the axon to reach the axon terminal. At electric synapses, the output is the electrical signal transmitted over the axon while at chemical synapses, the output is a neurotransmitter. Neurons are either sensory, motor or inter-neuron. The first type conveys sensory information. The second type conveys motor information. The third type conveys information between neurons.

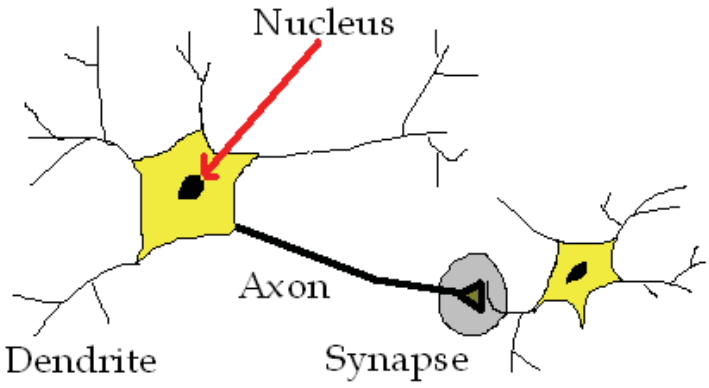


Fig. 2. Structure of a neuron network in the brain

An artificial neuron models a real neuron as depicted in Fig. 3. First, electric signals from other neurons are multiplied by weights (represented by the rectangles in Fig. 3) and then are input into the artificial neuron. The weighted signal values are them summed by an adder ("Σ" in Fig. 3) and the sum is subjected to a transfer function ("T" in Fig. 3) which is one of: i. linear, where the output is proportional to the weighted sum of inputs; ii. threshold, where the output is one of two values based on whether the weighted sum is greater or smaller than the threshold value; and iii. sigmoid, a non-linear function which most closely mimicks real neurons. Artificial neural networks are composed of several artificial neurons as a real neuron network is composed of many real neurons. Artificial neural networks come in different forms and shapes.

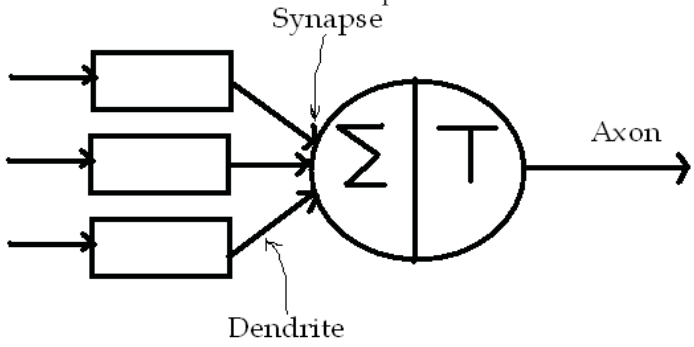


Fig. 3. Artificial neuron model

Artificial neural network organizations are either feedforward, or recurrent. Feedforward networks do not have cycles and signals flow from input to output. Recurrent neural networks have loops, i.e. links to neurons in the same or previous layers. The MLP neural network organization, shown in Fig. 4, is an example of feedforward network. Hebbian networks are example of recurrent networks. Recurrent networks because of their feedback

cycles go through dynamic state changes until the network stabilizes. Neural networks can be either fixed or adaptive. Fixed networks have unchanging weights usually set at the start and need not change as the network is expected to keep operating in the same way. Adaptive neural networks as those which allow their weights to change (i.e. allow the network to adapt and learn). Adaptive neural networks can therefore learn and their learning usually falls under two large classes: supervised or unsupervised. Supervised learning involves a supervisor who tells the network how to learn, i.e. what the network output should be for each input combination. In the supervised learning, the inputs and corresponding outputs are known, so the neural network learns by applying a "cost" function to generate the desired outputs for each input combination. Popular cost functions include the mean squared error function, and random functions. The mean squared error function attempts to minimize the error between the actual output value computed by the network and the desired output value. In unsupervised learning, the network is not told what the generated output should be for each input combination but the neural network learns by itself by self-organizing the data and identifying the data's characteristics and properties. Yet another class of learning is reinforcement learning. Reinforcement learning differs from the supervised learning problem in that correct data inputs and matching output are never presented to the network. In addition, sub-optimal actions are not explicitly corrected. Instead, in reinforcement learning, agents interact with the environment and supply the data to the network which attempts to formulate a policy for agent actions which optimizes some cost function. Evolutionary algorithms and simulated annealing are examples of other types of neural network training methods.

The MLP is an example of feedforward artificial neural network with multiple layers and where each neuron output in one layer feeds as input to the neurons in the next layer as shown in Fig. 4. A radial basis function (RBF) neural network, pictured in Fig. 5, is a 3-layer network where the output is the weighted basis function (usually Gaussian function) of the Euclidian distance of the input vector and the neuron's center vector.

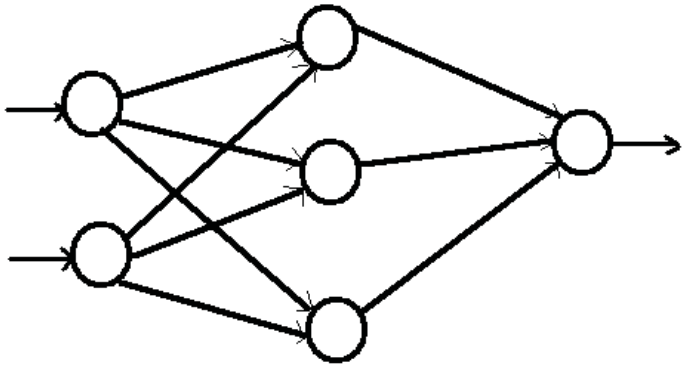


Fig. 4. MLP neural network

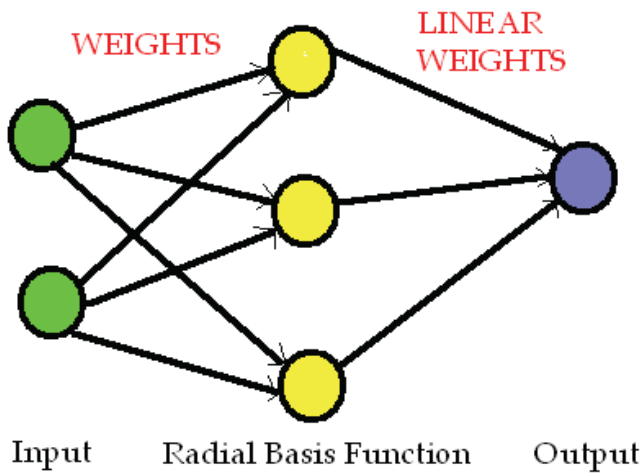


Fig. 5. RBF neural network

We chose our artificial neural network for secure password detection of the feed-forward type due to its simplicity and its suitability for this application.
We also employ the backpropagation algorithm for supervised training our network, a well known and widely used algorithm. The training algorithm minimizes the error between the obtained output and the required target output by finding the lowest point or minimum in the error surface. Starting with initial weights and thresholds, the training algorithms looks for the global minimum of the error surface. Usually the slope of the error surface at the present location and guides the next move down. It is the goal of the training algorithm to reach a low point on the surface which happens to be the local minimum, and in some unlucky situations, the algorithm stops at a local minimum. In the backpropagation algorithm, training moves the state along the steepest descent, reducing in each training epoch the error. The size of the move during each epoch is a function of the learning rate and the gradient (slope). At the end of each epoch, the weights are adjusted as a function of the error and the error surface gradient. The derivatives of the weights are computed to determine how the error changes as a result of increasing or decreasing the weights. The number of epochs taken to conduct training is determined by the error reaching a satisfactory value, or when the number of epochs reaches a predetermined number. Alternatively, training ends when the error stops shrinking.
Other training algorithms exist such as conjugate gradient descent --where after minimizing along one direction on the error surface, next moves assume that the second derivative along the first direction is held at zero-- and Quasi-Newton training (Bishop, 1995).

3. Previous work

Previous work on the subject include the work of (Reyhani & Mahdavi, 2007) who used a Radial Basis Function neural network and hash the (user ID, passwords) with a one-way hash function and then encrypt them. The authors claim that RBF trains much faster than with backpropagation learning (30x for a 200-sample training set and user ID and password

hashed to 13 characters) as RBF networks allow the selection of parameters for the hidden layer without the need for their optimization. Other recent work on the subject can be found in (Ciaramella et al, 2006) and (Wang & Wang, 2008) who used a Hopfield network, and (Li, Lin, & Hwang, 2001) who used a multilayer neural network with back-propagation training. (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008) both proposed authentication based on neural networks. Both (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008)'s solution are composed of two phases: a registration phase and a authorization phase. (Wang & Wang, 2008) allows a 3rd phase to allow subsequent password change.

In the registration phase, the neural network is trained to recognize a (user ID, password) combination. In both (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008), the user ID and passwords supplied by the user are encrypted before been fed to the neural network. (Wang & Wang, 2008) additionally sparsely-encodes the encrypted data with Reed Solomon code.

In the authorization phase, the user supplies the user ID and a password which are then encrypted and subsequently fed as input to the neural network. The network then processes this data and generates an output consisting of the encrypted password. If the two encrypted passwords --one generated by the neural network and the other supplied by the user in unecrypted form—match, then the password entered by the user matches the user ID supplied by the user and the access is authorized. Otherwise, there is no match between the user-supplied ID and password and the access is denied.

In both (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008), after neural network processing completes, the system must compare the ANN-generated password to the one provided by the user and determine if there is a match or not. This increases system vulnerability as the password generated by the neural network must be transferred to hardware which performs comparison. Although the password is encrypted, the password can be decrypted if the key is compromised, or the encrypted password can be saved on external storage and injected in the circuit (e.g. input of comparison hardware) in the future to gain illegal access.

A better approach which we employ is to let the neural network perform the comparison itself without the need for the neural network to generate a password and then transfer it to a comparator. In this approach, the neural network "memorizes" the combination of user ID and password during training and indirectly "codes" that information in its weights by enforcements and inhibitions and directly and internally performs the comparison with the user-supplied password without the need to transfer the 2 passwords for comparison by an entity external to the neural network. In that case, the output neuron generates a 1 for a (user ID, password) match or access authorization, or a 0 for access denial. In this neural network-based approach, the table of weights has to store the weights of the neural network and store new ones when the password is changed or new users are given permission to access the protected and secured valuables. Yet, the weights may not mean much to any one if the topology of the neural network (type of network, number of layers, number of neurons per layer, …) and weight format (order or weights and exact mapping of stored values into each weight variable) is kept secret. In addition as in (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008), the user ID and password combination can be immediately encrypted before being fed to the neural network and the neural network can be trained with those encrypted values to match the encrypted password. This is valuable if the distance between the keypad and the neural network is long necessitating transfer by Ethernet cables or other communication cables. Yet in either case whether the keypad-to-

neural network distance is long or short (both housed in same box), the password can be compromised if the box is bugged with a hardware sniffer to record and transmit keystrokes. One effective solution to this problem is to combine passwords or PINs with biometric data for a larger solution cost. Biometric techniques can render useless sniffing techniques employing vibration sensing (resulting from keystroke) or monitoring ground lines but also have their weaknesses. In the remainder of this section, we will assume that the keypad is safe and that keystrokes are neither transmitted nor recorded.
(Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008) report that the training time of multi-layer networks is long. When a new user ID and password combination requires recognition from the neural network, the neural network must be retrained again. However although RBF and Hopfield neural networks can be trained faster than multi-layer feedforward networks, the training time of multi-layer feedforward networks is often acceptable.

4. Artificial Neural Network Organization

In our artificial neural network for secure password detection, we chose the number of neuron layers to be 3, one for the input layer neurons which hold the input values, one for the hidden layer neurons which hold some of the key network's functionality, and one output layer neuron to generate the match/no match result. We later experimented with the number of hidden layer to determine which number of hidden layers results in the best match accuracy.
The number of neurons in the input layer is determined by i. the sizes (in characters) of the user ID and password; ii. the coding type of the user ID and password information, whether with binary values or with analog ones.
The number of neurons in the middle layer is also affected by the above network attributes as well as the connections between the input layer neurons and the middle layer numbers, their number and their origins and destinations.
In our experiment, we have chosen to limit our user ID and password each to 5 characters each, where each character is the A-H letter range, where the case is ignored, meaning that upper and lower case versions of the same letter are identical for all practical purposes. Thus each user ID has 1 matching password out of 85=32,768 possible password combinations of 8 letters.
Fig. 6 shows our artificial neuron organization assuming that each input user ID and password character is digitally coded by three ($\log_2 8$) binary values. Note that the top row represents the 15 input neurons each representing one of the 3x5=15 input user ID binary values (3 neurons per character), and the 15 input neurons below the top row represent the 15 input password binary values in the same input layer but drawn below for better presentation. The hidden layer shows 30 neurons, as many as input layer neuron, with each input layer neuron feeding into each of the hidden layer neurons. The bottom row shows the single output layer neuron fed from each neuron in the hidden layer.
When the user ID and password characters are represented by analog values, each analog value can be input to a single neuron, so the version of Fig. 6 with analog data values contains 10 neurons in the input layer, 10 in the hidden layer and 1 in the output layer. Again, all input layer neurons feed into each of the 10 hidden layer neurons which feed in turn into the single output layer neuron.

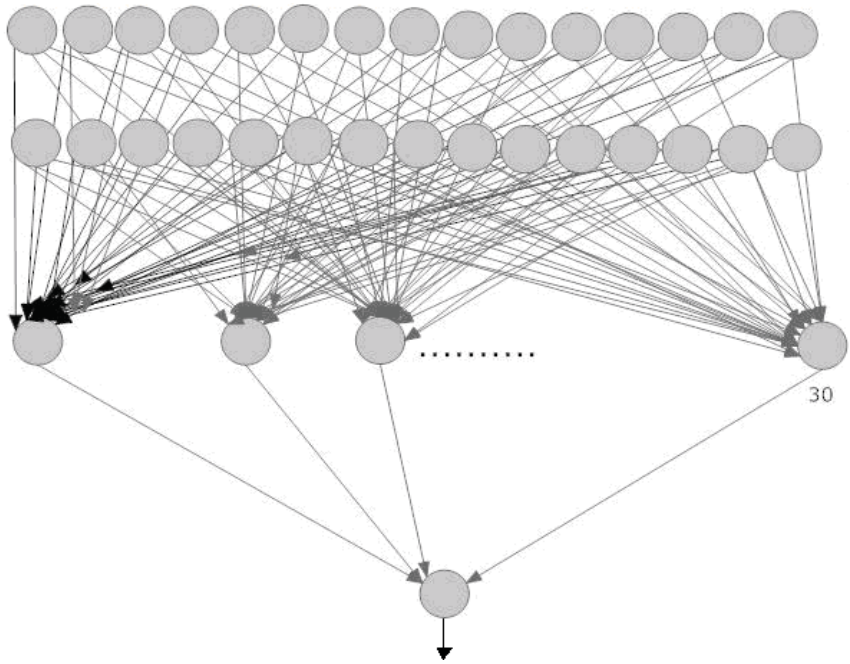


Fig. 6. Artificial neural network organization.

5. Data Coding

In the analog coding of the data, each input user ID and password character takes on a real number according to Table 1. Thus character "A" or "a" takes any value between 0 and 0.4. "B" is represented by all values in the range 0.5-0.9 and so on. Numbers in the boundary gaps, e.g. 0.4-0.5, are not intended to represent any useful characters.

Character	Ranges
A	0 - 0.4
B	0.5 - 0.9
C	1.0 – 1.4
D	1.5 – 1.9
E	2.0 – 2.4
F	2.5 – 2.9
G	3.0 – 3.4
H	3.5 – 3.9

Table 1. Real number coding

In the binary data coding version, each input user ID and password character in the range A-H is represented by 3 bits according to the coding convention of Table 2.
One major strength of real number (analog) coding is its cost saving as it represents each input character by a single neuron. Its major weakness is that characters at the bottom such

as “G’ and “H” are assigned larger values than characters on the top such as “A” and “B”. This is problematic because input values are multiplied by link weight values before being summed by the 2nd and 3rd layer neurons. Thus inputs with larger values result in larger products (than inputs with smaller values) which could be wrongly misinterpreted by the circuit as a strong match, and must be inhibited by the next layers in order to reduce their message strengths. Also smaller products are diluted along large product values when summed by a neuron.
This weakness is avoided in the binary coding where the character is represented by three independent and separate bits. The binary coding also removes the fuzziness between analog code boundaries such as 0-0.4 and 0.5-0.9 above, as values which fall into the boundary areas 0.4-0.5 could be either interpreted as an “A” or a “B.” With 0 and 1 digits, the likelihood of getting into these kind of debates is greatly reduced.

Character	Binary value
A	"000"
B	"001"
C	"010"
D	"011"
E	"100"
F	"101"
G	"110"
H	"111"

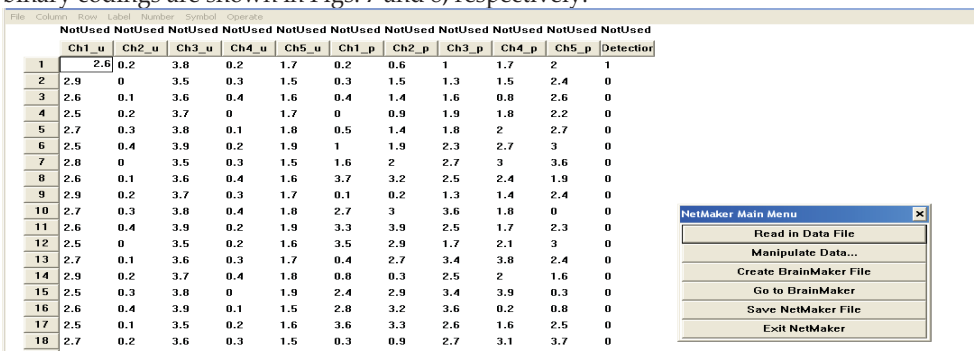
Table 2. Binary coding

Compared to real number coding, digital coding requires more neurons. Compared to other digitl codigns, one strength of the selected binary coding of Table 2 is that it simplies the representation of each character by the least number of bits, rather than, say, to enhance its fault tolerance snd/or sparseness as in 1-hot assignment and other encoding schemes. Other coding techniques based on Hamming, Reed Solomon, and others could be also used. We just chose the simplest coding.

6. Training, Simulation and Test

We used the BrainMaker (IEEE, 1992) application for training and simulating our network and retrieving match accuracy results. BrainMaker is a popular neural network simulation package and has been used extensively by researchers such as (Dombi & Lawrence, 1994). We created a list of 1000 (user ID, password, output) triplets where only one triplet had an output of 1 (means match) for the matching (user ID, password combination). We focused on a single user ID so all the user ID values in all 100 triplets were identical. By providing the output match/no-match value during training, we avoided the need to compare the

memorized password from the entered password as in (Reyhani & Mahdavi, 2007) and (Wang & Wang, 2008). The password values varied slightly to a great extent from the matching password value. In the real number coding experiment we assigned values to the user ID and password characters as in Table 1 and we simulated a network of 10 input neurons and 1 hidden layer neuron and 1 output neuron. In the binary coding experiment, we assigned values to the user ID and password characters as in Table 2 and we simulated a network of 30 input neurons and 30 hidden layer neurons and 1 output neuron. The created training set of 1000 triplets or lines was first written into an Excel spreadsheet and then we changed that file's extension to ".dat." We then started the NetMaker application to read the .dat file and to create from it the BrainMaker files including the definition file, training fact file and the testing fact file. The resulting training set in NetMaker format for analog and binary codings are shown in Figs. 7 and 8, respectively.



File Column Row Label Number Symbol Operate

	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed
	Ch1_u	Ch2_u	Ch3_u	Ch4_u	Ch5_u	Ch1_p	Ch2_p	Ch3_p	Ch4_p	Ch5_p	Detectior
1	2.6	0.2	3.8	0.2	1.7	0.2	0.6	1	1.7	2	1
2	2.9	0	3.5	0.3	1.5	0.3	1.5	1.3	1.5	2.4	0
3	2.6	0.1	3.6	0.4	1.6	0.4	1.4	1.6	0.8	2.6	0
4	2.5	0.2	3.7	0	1.7	0	0.9	1.9	1.8	2.2	0
5	2.7	0.3	3.8	0.1	1.8	0.5	1.4	1.8	2	2.7	0
6	2.5	0.4	3.9	0.2	1.9	1	1.9	2.3	2.7	3	0
7	2.8	0	3.5	0.3	1.5	1.6	2	2.7	3	3.6	0
8	2.6	0.1	3.6	0.4	1.6	3.7	3.2	2.5	2.4	1.9	0
9	2.9	0.2	3.7	0.3	1.7	0.1	0.2	1.3	1.4	2.4	0
10	2.7	0.3	3.8	0.4	1.8	2.7	3	3.6	1.8	0	0
11	2.6	0.4	3.9	0.2	1.9	3.3	3.9	2.5	1.7	2.3	0
12	2.5	0	3.5	0.2	1.6	3.5	2.9	1.7	2.1	3	0
13	2.7	0.1	3.6	0.3	1.7	0.4	2.7	3.4	3.8	2.4	0
14	2.9	0.2	3.7	0.4	1.8	0.8	0.3	2.5	2	1.6	0
15	2.5	0.3	3.8	0	1.9	2.4	2.9	3.4	3.9	0.3	0
16	2.6	0.4	3.9	0.1	1.5	2.8	3.2	3.6	0.2	0.8	0
17	2.5	0.1	3.5	0.2	1.6	3.6	3.3	2.6	1.6	2.5	0
18	2.7	0.2	3.6	0.3	1.5	0.3	0.9	2.7	3.1	3.7	0

Fig. 7. Sample real number-coded training data in NetMaker format

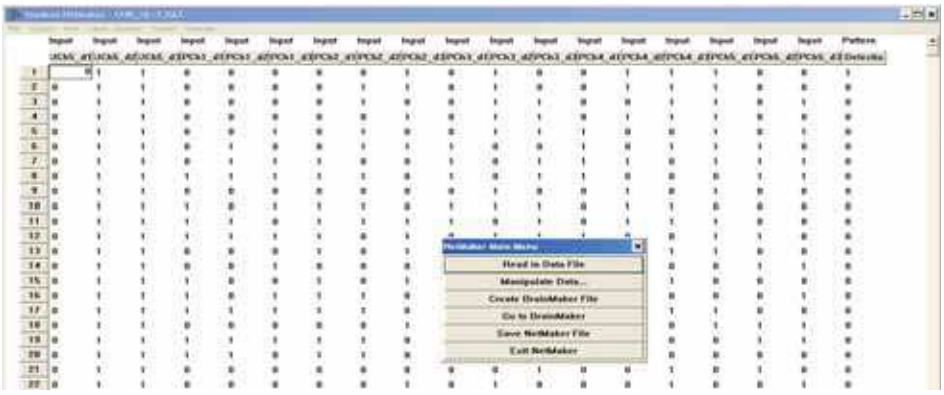


Fig. 8. Sample binary-coded training data in NetMaker format

A snapshot of the BrainMaker files and their extensions is shown in Fig. 9.

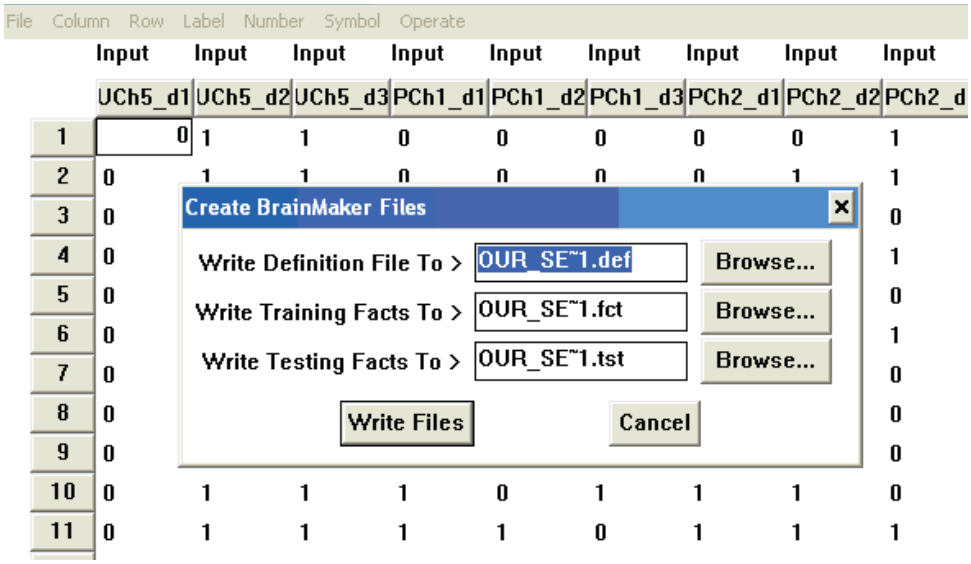


Fig. 9. Associated BrainMaker files

The BrainMaker application was then launched to train, test, manipulate data, and run network simulations. During the training phase, we could see the changes that occurred to the output in graph, numeric or histogram formats, allowing us to immediately discover how big a training set was necessary to reach the output neuron's password match accuracy, or if the simulated network with its tested parameters will satisfy our match accuracy requirements.

Out of the 1000 lines in the training set, we reserved 90% or 900 for training the network and 10% or 100 for testing the network after it has been trained for password detection accuracy, as illustrated in Fig. 10. The 10% reserved for testing are not used at all for training the network and are exclusively used for testing the network, i.e. checking that it generates a correct output, 0 when the password does not match the user ID, and 1 otherwise.

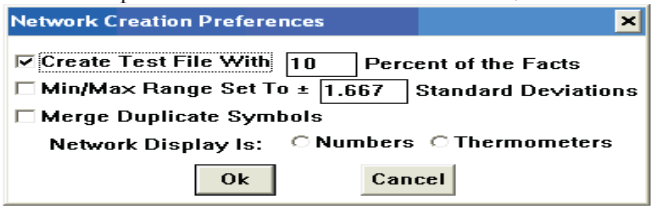


Fig. 10. Creating test set

7. Results

After training the network with real number coded inputs with the BrainMaker simulator and with a training set of 900 lines, we obtained 1.44% incorrect detections and 98.5% good password detections. The learn and tolerance values were 0.085 and 0.1, respectively. Next, we switched to binary coded inputs, and at the end of training, the accuracy stood at 99.7%

good detections and 0.0022% bad or incorrect detections. In the testing phase, we tested the network with binary-coded inputs with the 100 lines and obtained a perfect 100% accuracy. Attempting to validate these accuracy results and discovering any correlation to the size of the testing set, we increased the testing file to 50% (500 lines or test cases) instead of 10%. The accuracy results did not change which means that our artificial neural network has learnt well and has been sufficiently trained.

7.1 Experimenting with number of hidden layers and number of neurons in the hidden layer

Afterwards, we experimented with the number of hidden layers and adding noise as illustrated in Fig. 11. By doubling the number of hidden layers to 2, we got 11 incorrect detections, and thus more hidden layers reduced the accuracy. By increasing the number of neurons in the hidden layer from 30 to 50 neurons, the password detection accuracy also decreased. We got 2 bad values (false matches) only when the hidden layer had 30 neurons, but after increasing the number of neurons in the hidden layer we got 12 bad values. These last two experiments caused our network to overlearn. Extra added neurons or hidden layers of neurons falsely twisted the results. This is similar to asking the opinions of two or more persons on some subject when they have unidentical opinions, and then somewhat averaging their answers in order to make a judgment, rather than taking the advice of a fewer number of people but who are in accord.

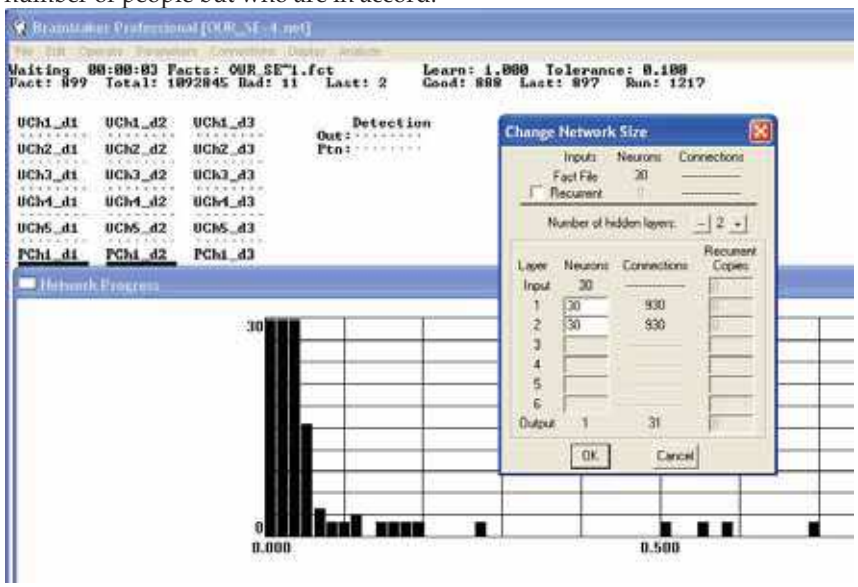


Fig. 11. Changing the artificial neural network size (number of hidden layers, and number of neurons per hidden layer)

7.2 Adding noise to the network

Moreover, we added noise to the network (see Fig. 12) and observed that as expected, the more noise was added the more the results worsened. In fact, as a result of noise addition,

the accuracy dropped to 96% and the number of bad result increased to 28 bad values which mean 3.11% incorrect detections. Noise causes the data values to increase or decrease from their original values and throws the detection accuracy off but it is important for the artificial neural network to keep operating at high accuracies in the presence of noise.

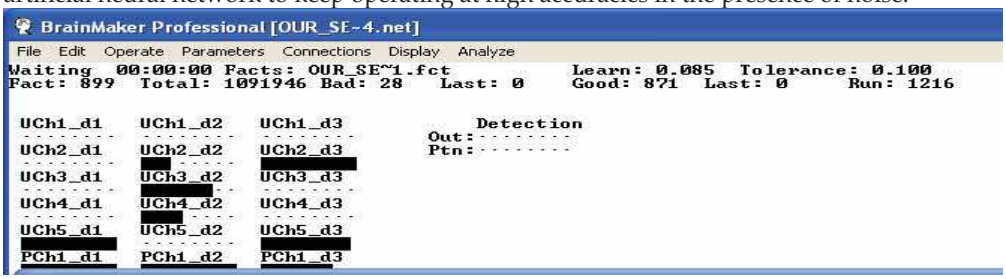


Fig. 12. Adding noise

The results of all our experiments with 1000 (user ID, password) combinations in total, of which 900 are used for training and 100 for testing, are summarized in Table 3.

ANN EXPERIMENT	Accuracy
Default (with binary coded inputs, one hidden layer & one input layer of 30 neurons each)	100% (after testing phase)
Binary coding with 2 hidden layers	97.6 %
Binary coding with added noise	96 %
Binary Coding, with number of neurons in the hidden layer increased to 50 neurons	98.2 %
Binary Coding, with number of neurons in the hidden layer reduced to 6 neurons	95.6 %
With real number coded inputs	98.4 %

Table 3. Accuracy results

8. Conclusion

Artificial neural networks are used in many applications ranging from control, to pattern recognition, to classification. In this Chapter, we have described our experience in designing artificial neural networks for secure access authorization. We designed our feedforward network with 3 layers, created the training set and trained our network with the

backpropagation algorithm. We then simulated and tested our design with BrainMaker and collected accuracy results. We were able to achieve 100% result accuracy with our network. Increasing the number of hidden layers or the number of neurons in the hidden layer or adding noise all individually hurt our network's password detection accuracy.
Potential future work is to extend the network to accept alphanumeric inputs and extend the input range to A-Z while keeping the number of neurons under control.

9. Acknowledgments

We acknowledge Dr. Azam Beg of the College of Information Technology, UAE University for useful discussions.

10. References

Bishop, C. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press, UK.

Ciaramella, A.; D'Arco, P., De Santis, A., Galdi, G. & Tagliaferri, R. (2006). Neural Network Techniques for Proactive Password Checking. *IEEE Trans. on Dependable and Secure Computing*, Vol. 3, No. 4, pp. 327-339.

Dombi, G. & Lawrence, J. (1994). Analysis of Protein Transmembrane Helical Regions by a Neural Network. *Protein Science*, Vol. 3, (1994), pp. 557-566. Cambridge University Press.

Fausett, L. (1994). *Fundamentals of Neural Networks*, Prentice Hall, New York.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*, 2nd Ed, Prentice Hall, NY.

IEEE Expert Staff. (1992). Brainmaker Professional. *IEEE Expert: Intelligent Systems and Their Applications*, Vol. 7, No. 2, (April 1992), pp. 70-71.

Li, L.; Lin, I. & Hwang, M. (2001). A Remote password authentication scheme for multiserver architecture using neural networks. *IEEE Transactions on Neural Networks*, Vol. 12, No. 6, (November 2001), pp. 1498-1504.

Lin, I., Ou, H. & Hwang, M. (2005). A User authentication system using back-propagation network. *Neural Computing and Applications*, 14, (2005), pp. 243-249.

Reyhani, S. & Mahdavi, M. (2007). User Authentication Using Neural Network in Smart Home Networks. *International Journal of Smart Home*, Vol. 1, No. 2, (2007), pp. 147-154.

Rumelhart, D; McClelland, J. (Eds, 1986). *Parallel Distributed Processing: Explorations in the MicroStructure of Cognition*, MIT Press, Cambridge.

Wang, S. & Wang, H. (2008). Password Authentication Using Hopfield Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 38, No. 2, (March 2008), pp. 265-268.

Pattern Recognition
Edited by Peng-Yeng Yin

ISBN 978-953-307-014-8
Hard cover, 568 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

For more than 40 years, pattern recognition approaches are continuingly improving and have been used in an increasing number of areas with great success. This book discloses recent advances and new ideas in approaches and applications for pattern recognition. The 30 chapters selected in this book cover the major topics in pattern recognition. These chapters propose state-of-the-art approaches and cutting-edge research results. I could not thank enough to the contributions of the authors. This book would not have been possible without their support.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Fadi N. Sibai, Aaesha Shehhi, Sheikha Shehhi, Buthaina Shehhi and Najlaa Salami (2009). Designing and Training Feed-Forward Artificial Neural Networks For Secure Access Authorization, Pattern Recognition, Peng-Yeng Yin (Ed.), ISBN: 978-953-307-014-8, InTech, Available from: http://www.intechopen.com/books/pattern-recognition/designing-and-training-feed-forward-artificial-neural-networks-for-secure-access-authorization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821