Authentication Algorithm for Intrusion Detection in Wireless Networks

[1]Dr. A.K. Santra, [2]Nagarajan S,
[1]Professor and Dean, MCA Department, CARE School of Computer Applications, Tiruchirappalli, Tamil Nadu.
[2]Research Scholar, Bharathiar University, Coimbatore and Professor and Head, The Oxford College of Science, Bangalore, Karnataka.

Abstract

Security has been a major issue in wireless networks. In a wireless network, wireless devices are prone to be unauthorized accessing data or resources. Hence it becomes necessary to consider issues of security such as : 1. Authentication, 2. Access Control. Traditional methods of Authentication has been to assign user names and passwords. This is extremely vulnerable to be accessed and misused. Therefore, to overcome these problems, This paper proposes a method by which the Username and Password are stored in an Hashed format, with the help of Neural Network learning the hashes instead of storing them in the tables. Thus, reducing the risk of being accessed. Therefore, a combination of authentication and access control could make a good tool for intrusion detection.

Key words:Authentication, Access Control, Fuzzy Art Map Neural Network, Dynamic Niche Particle Swarm Optimization, Wireless network, Fuzzy Logic

1. Introduction

User Authentication is an important aspect of allowing authorized users into the wireless or wired networks. This also helps to a large extent to do intrusion detection. Thereby minimizing the risk of intrusions.

Authentication is done mainly using the methods of passwords, biometrics and smart cards. [1].

Passwords have been a very convenient way to authenticate users. It has been the traditional, inexpensive and popular method to authenticate users. But, there are other methods suggested by various researchers. The most prominent among them identified in the consumer market are smart cards and biometrics which are more secure than the password method [2].

Even though passwords are convenient, there are a few disadvantages. This is due to the fact that many people have the inclination to choose decently short and simple passwords which are prone to the exhaustive search or dictionary attacks, as revealed in the papers already published [3] [4] [5].

For a password scheme, the user's username and password is stored in a table. In a scenario where intrusion takes place, the intruder can play havoc by using the passwords or altering it to their benefit. Therefore, passwords kept at a specified point may be prone to intrusions and alterations. Hence, this is a potential threat to the system and the table needs to be secured [5].

There are other schemes like encrypting the passwords and storing it. The table again needs to be stored in a secure place. As the intruder may not be in position to get the original password, But, can always substitute the pattern existing in the table, which comprises the security of the password.

There is an approach which uses Multi-layer perceptron (MLP) neural network to defeat this issue of tables [6]. Further, to this a similar scheme has been proposed in [5] using the Radial Basis Function neural Network.

In this paper, the scheme proposed is for a wireless network and a modification for the method proposed in [5]. The paper proposes to replace the use of the Radial Basis function Neural Network with the Fuzzy ARTMAP Neural Network to decide on any intruder at the time of gaining access to the system itself. Also training time is less than that of the RBF. Hence, the proposed method is more efficient than the method proposed in [5].

2. The proposed Authentication Scheme

The Proposed authentication system uses fuzzy ARTMAP Network. This Scheme works in two phases Viz., one the user registration and the second the user authentication phase. The scheme works by accepting a username and password from the user in the first part and in the second part the system would typically validate the user's authenticity.

Usernames	Passwords
UID_1	PWD_1
UID_2	PWD_2
.	.
.	.
.	.
UID_n	PWD_n

Usernames	Hashed Passwords
UID_1	$f(PWD_1)$
UID_2	$f(PWD_2)$
.	.
.	.
.	.
UID_n	$f(PWD_n)$

Figure 1[5]: Hashed user name and password database

2.1 The Fuzzy ARTMAP Neural Network

This paper uses the Fuzzy ARTMAP Neural Network. The Fuzzy ARTMAP is a feed forward Neural Network trained using a supervised learning algorithm.

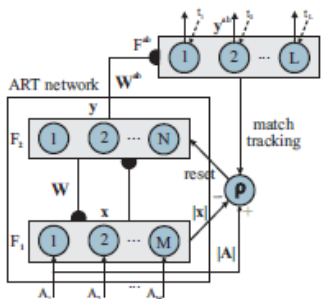

Figure:2 An Fuzzy ARTMAP neural network architecture specialized for pattern classification [22].

2.2 Supervised Training of Fuzzy ARTMAP

2.2.1 The Fuzzy ARTMAP Neural Network

ARTMAP refers to a family of neural network architectures based on Adaptive Resonance Theory (ART) [10] that is capable of fast, stable, on-line, unsupervised or supervised, incremental learning, classification, and prediction [12, 13]. ARTMAP is often applied using the simplified version shown in Figure 2. It is obtained by combining an ART unsupervised neural network [10] with a map field. The ARTMAP architecture called fuzzy ARTMAP [13] can process both analog and binary-valued input patterns by employing fuzzy ART [11] as the ART network.

The fuzzy ART neural network consists of two fully connected layers of nodes: an M node input layer, F_1, and an N node competitive layer, F_2. A set of real-valued weights $\boldsymbol{W}=\{w_{ji}\in[0,1]:i=1,2,...,M;j=1,2,..,N\}$ is associated with the F_1 -to- F_2 layer connections. Each F_2 node j represents a recognition category that learns a prototype vector $\boldsymbol{w}_j^{ab}=(\boldsymbol{w}_{j1}^{ab}, \boldsymbol{w}_{j2}^{ab}, \ldots \boldsymbol{w}_{jL}^{ab})$. The F^2layer of fuzzy ART is connected, through learned associative links, to an L node map field F^{ab}, where L is the number of classes in the output space. A set of binary weights $W^{ab}=\{w_{jk}^{ab}\in[0,1]:j=1,2,...,N;k=1,2,...,L\}$ is associated with the F_2 -to-F^{ab} connections. The vector $W_j^{ab}=(W_{j1}^{ab}\ W_{j2}^{ab}\ldots\ldots\ W_{jL}^{ab})$ links F_2 node j to one of the L output classes.

In batch supervised training mode, ARTMAP classifiers learn an arbitrary mapping between training set patterns $\boldsymbol{a}=\{a_1,a_2,\ldots,a_m)$ and their corresponding binary supervision patterns $\boldsymbol{t}=\{t_1,t_2,\ldots,t_L)$. These patterns are coded to have unit value $t_K=1$ if K is the target class label for $\boldsymbol{a}$, and zero elsewhere. Algorithm 1 describes fuzzy ARTMAP learning.

Algorithm 1 Fuzzy ARTMAP learning.

1. Initialization—All the F_2 nodes are uncommitted, all weight values W_{ij} are initialized to 1, and all weight values W_{jk}^{ab} are set to 0. An F_2 node becomes committed when it is selected to code an input vector **a**, and is then linked to an F^{ab} node.

Values of the learning rate βЄ[O,l], the choice α>0 , the match tracking 0<ε<<1, and the baseline vigilance ρ-Є[0,1] parameters are set.

2. Input pattern coding—When a training pair (**a** ,**t**) is presented to the network undergoes a transformation called complement coding, which doubles its number of components. The complement-coded input pattern has M=2m dimensions and is defined by **A**={**a**,**a**^{c})={a_1,a_2,…,a_m; a^c_1,a^c_2,…,a^c_m), where a^c_1;=(l-a,.), and a_1Є[0,l]. The vigilance parameter p is reset to its baseline value p$^-$.

3. Prototype selection-pattern **A** activates layer F_l and is propagated through weighted connections **W** to layer F_2. Activation of each node j in the F_2 layer is determined by the Weber law choice function:

$$T_j(A)=\frac{|A\wedge W_j|}{\alpha+|W_j|} \qquad (1)$$

where \.\ is the L_1 norm operator defined by |**w**$_j$|=$\Sigma_{i=1}^{M}$ \w$_{tj}$\ , □ is the fuzzy AND operator, (**A**^w,), and a is the user-defined choice parameter. The F_2 layer produces a binary, winner-take-all pattern of activity **y**={y_l,y_2,-,y_N) such that only the node j=J with the greatest activation value J=argmax{7y7 = 1,2,..., iV] remains active; thus y.,= l and y^OJ^J. If more than one T_i is maximal, the node j with the smallest index is chosen. Node J propagates its top-down expectation, or prototype vector Wj, back onto F_1 and the vigilance test is performed. This test compares the degree of match between **w**., and **A** against the dimensionless vigilance parameter ρ Є [0,l] :

$$\frac{|A\wedge W_J|}{|A|}=\frac{|A\wedge W_J|}{M}\geq\rho \qquad (2)$$

If the test is passed, then node J remains active and resonance is said to occur. Otherwise, the network inhibits the active F_2 node (i.e., T, is set to 0 until the network is presented with the next training pair (**a**,**t**)) and searches for another node J that passes the vigilance test. If such a node does not exist, an uncommitted F_2 node becomes active and undergoes learning (Step 5). The depth of search attained before an uncommitted node is selected is determined by the choice parameter a .

4. Class predictions pattern **t** is fed directly to the map field F^{ab}, while the F_2 category learns to activate the map field via associative weights $\mathbf{W}^{ab}$. The F^{ab} layer produces a binary pattern of activity $\mathbf{y}^{ab}$={y_1^{ab} , y_2^{ab} .., y_L^{ab})=**t** ^$\mathbf{w}_j^{ab}$ in which the most active F^{ab} node K=argmax{y^{ab}_K:k=1,2,...,L) yields the class prediction (K=k(J)). If node K constitutes an incorrect class prediction, then a match tracking signal raises the vigilance parameter ρ just enough:

$$\rho=\frac{|A\wedge W_J|}{M}+\varepsilon \qquad (3)$$

where ε=0$^+$, to induce another search among F_2 nodes in Step 3. This search continues until either an uncommitted F_2 node becomes active (and learning directly ensues in Step 5), or a node J that has previously learned the correct class prediction K becomes active.

5. Learning-Learning input **a** involves updating prototype vector $\mathbf{w}_j$, and, if J corresponds to a newly-committed node, creating an associative link to F^{ab}. The prototype vector of F_2 node J is updated according to:

$$W_j = \beta(A\wedge W_j)+(1-\beta)W_j \qquad (4)$$

where β is a fixed learning rate parameter. The algorithm can be set to slow learning with 0<β<1, or to fast learning with β=1 . With complement coding and fast learning, fuzzy ARTMAP represents category j as an m-dimensional hyperrectangle R_j that is just large enough to enclose the cluster of training set patterns **a** to which it has been assigned. That is, an M -dimensional prototype vector $\mathbf{w}_j$ records the largest and smallest component values of training subset patterns **a** assigned to category j. The vigilance test limits the growth of hyperrectangles – a ρ close to 1 yields small hyperrectangles, while a ρ close to 0 allows large hyperrectangles. A new association between F_2 node J and F^{ab} node K (k(J)=K) is learned by setting $w_{jk}{}^{ab}$=1 for k = K, where K is the target class label for **a**, and 0 otherwise. The next training subset pair {**a**, **t**) is resented to the network in Step 2.

Batch supervised training ends in accordance with some learning strategy, following one or more

epochs. An epoch is defined as one complete presentation of all the patterns of a finite training data set. Once the weights $\boldsymbol{W}$ and $\boldsymbol{W}^{ab}$ have been found through this process, ARTMAP can predict a class label for an input pattern by performing Steps 2, 3 and 4 without any vigilance or match tests. During testing, a pattern $\boldsymbol{a}$ that activates node J is predicted to belong to class $K=k(J)$. The time complexity required to process one input pattern, during either a training or testing phase, is $O(MN)$.

2.2.2 Typical Batch Supervised Learning Strategies

Given a large training data set, one of the following four learning strategies are typically used to select e, the total number of epochs needed to end batch supervised learning by fuzzy ARTMAP:

- *One epoch* (1EP):
  The learning phase ends after one epoch ($e=1$) of the training data set. (This learning strategy is mainly used for reference during simulations.)
- *Convergence based on training set classifications* (CONVp):
  The learning phase ends after the epoch e for which all patterns of the training data set have been correctly classified by the network. Convergence occurs when $\Sigma_l(t_l-y^{ab}_{l,e})=0$ over all patterns l in the training set. Note that this strategy is prone to convergence problems for data with overlapping class distributions, as some training patterns in the overlap region may never be correctly classified.
- *Convergence based on weight values* (CONVw):
  The learning phase ends after the epoch e for which the weight values have converged. Convergence occurs when the sum-squared-fractional-change (SSFC) of weights $\boldsymbol{W}$ for a two successive epochs, $e-1$ and e, is less than 0.001, $SSFC=\Sigma_j\Sigma_i.(W_{ij}(e)-W_{ij}(e-1))^2<0.001$.
- *Hold-out validation* (HV):
  The learning phase ends after the epoch e for which the generalization error is minimized on an independent validation subset. Learning is performed using a holdout validation technique [15], with network training halted for validation after each epoch. In practice, the number of epochs that achieves the lowest generalization error should be selected by observing several epochs after e to avoid falling into local minimums. With large data sets considered in this paper, the HV is an appropriate learning strategy. If data was limited, k-fold cross-validation would be a more suitable validation strategy, at the expense of some estimation bias due to crossing. [14]

2.2.3 Dynamic Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique that was inspired by social behavior of bird flocking or fish schooling. With PSO, each particle corresponds to a single solution in the optimization space, and the population of particles is called a swarm. Particles move through the optimization space and change their course under the guidance of a cognitive influence (i.e., their own previous search experience) and a social influence (i.e., their neighborhood previous search experience). Unlike evolutionary algorithms (such as genetic algorithms), each particle always keep in memory its best position and the best position of its surrounding.

During incremental learning of new data blocks D_t, the FAM hyperparameters must be adjusted to maximize the classification rate. It has been shown that this adaptation corresponds to a dynamic optimization problem such as

$$\text{maximize } \{f(h, t) \mid h \in R^4, t \in N_i\} \qquad (4)$$

where $f(h,t)$ is the classification rate of FAM for a given vector of hyperparameters $h = (\alpha,\beta,\varepsilon, p)$, after learning data set D_t and at a discreet time t [16]. Hence, a dynamic version of the original PSO algorithm is needed to properly track optimal system parameters through time.

Originally developed for static optimization problems, the PSO algorithm has been adapted for dynamic optimization problems adding mechanisms to (1) modify the social influence to maintain diversity in the optimization space and detect several optima, (2) detect changes in the objective function by using the memory of each particle, and (3) adapt the memory of its population if change occur in the optimization environment. The latest particle swarm optimization algorithms developed to insure diversity in the swarm are presented in [18], [19], [20], [21]. Change detection and memory adjustment mechanisms for DPSO are presented in [22], [23], [24], [25].

In this paper, the adaptive classification system

(ACS) uses a Dynamical Niching PSO (DNPSO) [18] to maximize FAM classification rate as a function of its hyperparameters vector h = *(α,β,ε, p)* when learning new data blocks D_t (Figure 1). The optimization space is defined by the four FAM hyperparameters *(α*Є[0.001,100], *β*Є [0,1], εЄ [— 1,1], and *p* Є [0,1]. At each DNPSO iteration T, the objective function of particle *n,* /(h_n(ד),t), is defined by the classification rate of the FAM network corresponding to particle *n, FAM_n,* evaluated at that particle's position, h_n(ד).

The DNPSO algorithm adapted to dynamic optimization has been shown to converge toward global maximum in a multimodal type III optimization environment, where both location and value of optima points change, with the moving peaks benchmark [18]. It maintains diversity (1) by using a local neighborhood topology, in which subswarms are dynamically created around master particles, that are their own best position amongst their neighborhood, and (2) by allowing free particles that do not belong to any any subswarms, to move independently. DNPSO was adapted for dynamic optimization problems by simply updating the fitness of their best position f(h*,t) at each iteration. Normally, this would double the number objective function evaluations, leading to a very costly process. However, for an ACS, changes in the objective function only occur when a new data block D_t becomes available. Thus, the fitness of the best particle positions is only

Algorithm 1 DPSO-based learning strategy for the ACS using a swarm of FAM networks.
Inputs: New data sets D_t for learning.
Outputs: The *N* best vectors h and networks *FAM.*

1: Initialization:
- initialize all *N* networks FAM_n and $FAM^{s\wedge rt}$,
- set the swarm parameters,
- set PSO iteration counter at T = 0, and
- randomly initialize particles positions and velocities.

Upon reception of a new data block D_t, the following-incremental process is initiated:

2: for each particle *n,* where $1 \leq n \leq N$ do
3: Training of FAM_n with validation using D\ and D^v, and f/(h_n*,t) estimation using D^f.
4: end for
5: while PSO did not reach stopping condition do
6: Update particle positions according to the DNPSO algorithm.
7: for each particle *n,* where $1 < n < N$ do
8: $FAM_{est} \leftarrow FAM^{s\wedge rt}$
9: Training of FAM_{est} with validation using D^t and D^v, and /(h_n(r),t) estimation using D^f.
10: if f(h_n(r),t) > f(h*,t) then
11:{h_n*, FAM_n, *f*(h_n*,t)} ←{h,,(r), FAM_{est}, *f*(h_n(*t*), *t*)}
12:end if
13: end for
14: T = T +
15: end while
16: for each particle *n,* where $1 \leq n \leq N$ do
17: $FAM_n^{start} \leftarrow FAM_n$
18: end for

updated when a new D_t is presented to the system, *before* the iterative DNPSO process.

D. Heterogeneous Ensemble of FAM Networks

Algorithm 1 describes the DPSO learning strategy for ensembles of FAM networks in dynamic classification environments. This algorithm supposes that the incremental process starts without any a priori knowledge of the class distributions Pfc(a). Given a new learning data block D_t, it cojointly optimizes the system parameters using a swarm with *N* particles, and stores *2N* + 1 FAM networks. Indeed, for each particle, the system stores *n = 1...N* networks $FAM^{s\ tart}$ in a short term memory to preserve the model associated with the best position of each particle (h*) at time £ — 1, and FAM_n, the model associated with h* during the optimization process at time *t.* It also stores FAM_{est}, a network employed for fitness estimation by the algorithm.

First, the initialization process takes place at line 1. All the neural networks are initialized, and the swarm's parameters are set. Each particle position is then randomly initialized within their allowed range.

When a new D_t becomes available, the optimization process begins. Fitness associated to each particle best position, /(h*, *t),* is updated according to the new data along with each network FAM_n (lines 2-4). Then, unless one of the stopping criteria is reached, the optimization process continues were it was previously halted (lines 5-15). The DNPSO algorithm defines the subswarms and free particles, and iteratively update each particle's new position along with their fitness (lines 6-13). If new personal best positions are found, the position, fitness, and network associated to the personal best *(FAM_n)* are updated (lines 10-12). In the cases of equality between /(h_n(r),t) and /(h*,£), simpler models are preferred. The position with the lighter network (the one with the less *Fi* nodes) then becomes h*. Finally, the iteration counter is incremented (line 14).

Once the DNPSO algorithm converges, the neural networks associated to each personal best *(FAM_n)* are

stored as FAM$^{s\ tart}$ (lines 16-18). Those networks will serve as a short term memory of the swarm's state time *t,* and for learning data block Dt+i. Each time a particle's fitness (i.e., classification rate) is estimated on D_t+*i,* the best network previously obtained at time *t,* saved in *FA*M$^{s\ t\wedge rt}$, serves as a starting state and is copied to FAM_{est} prior training.[3] Moreover, to minimize the impact of pattern presentation order on FAM, overall fitness is defined using the average classification rate of FAM trained on *D*\ over five different random pattern presentation orders, and FAM_{est} is the network that yielded the best classification rate.

This way, each particle is allowed to evolve according to its own knowledge of previous learned training data, and offers a different perspective of the problem. Thus, diversity is not only maintain for the particle positions, with the use of the DNPSO mechanism, but also for the models associated with each particle. By selecting a subset of networks amongst the swarm, the creation of an heterogeneous ensemble of classifiers [17] is possible. When used for class prediction, an ensemble of networks is directly selected from the swarm: one for each local best (including the global best), and completed with networks associated to the best free particles. A majority vote then decides the prediction. In the case of a tie, simpler models are again preferred, and the class predicted by the smallest networks (the ones that yielded the fewer *F2* nodes) is declared winner of the vote. [26]

3. The User Registration Phase

In this phase, the system administrator obtains the training patterns from usernames and passwords to train the neural network.

1. Every user will have to select a strong username and password, then pass it on to the system administrator.
2. The system applies the MD5 algorithm on the username and password provided. This output now becomes the required result for the training pattern.
3. The ACSII codes of the characters to be normalized before the training of the Neural Network.
4. The Fuzzy ARTMAP is used to train the patterns. Once the training process is completed the system administrator stores the ARTMAP network weights in the system as shown in Figure 3.

Password (desired output)	Username (input)
Example: 3377	Example: albert
Hashed Password	Hashed Username
AnX4H.obkufdM	zn25o23Zb6A48

Figure 3 [5]: Sample Training of the FAM

5. The User Authentication phase

The System uses the trained fuzzy ARTMAP Neural Network and applies the same MD5 hash function to give the authenticity of the user.
The authentication process is:

1. The system applies the same MD% function on the entered username and password.
2. The system will get the output from the trained Neural Networks.
3. The system will compare the output of the Fuzzy ARTMAP Neural Network with the hashed password. If the result matches then the user is given access, else, the user is registered the user is rejected as an intruder as in figure 4.

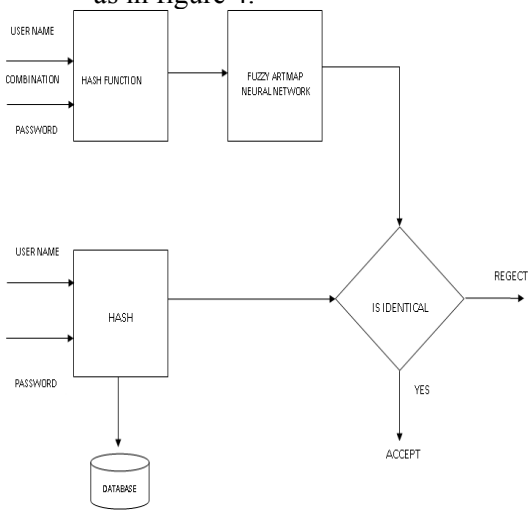

Figure 4. The user authentication phase in proposed system

6.Experimental Results

200 samples users are considered to train the fuzzy ARTMAP Neural Network. Each username and password will consist of 8-bits of characters. These usernames and passwords are hashed with MD5 and are transformed to the encrypted words that are used in the training. Each of them consists of 13 characters and the ACSII code of Therefore, the fuzzy ARTMAP Neural Network architecture has 13 input units in the input layer, 13 processing units in the hidden layer and 13 output units in the output

layer. When the training of the Fuzzy ARTMAP is completed the weight values of the Neural Network is stored in a 200 * 13 array. The array is then used to validate the authenticated users in the authentication phase. This system proposed can work on a P-IV at 2 GHz with a minimum of 256 MB size of RAM.

6.1 Accuracy and performance analysis

The proposed system has a good accuracy, as the trained Fuzzy ARTMAP generates outputs that are exactly the required outputs. When the user keys in the wrong username and password, the system inputs the hashed username and password to the trained Fuzzy ARTMAP Neural Network and gets a hashed password that is not identical to the right username and password. Therefore, the user is not given access as he is considered to be an intruder.

The training patterns to test the authentication. In this system when the right username and password is given, the output system is the corresponding hashed password. Each username and password are to be registered which is then converted to its corresponding hash using MD5. Further, this is used to train the Fuzzy ARTMAP Neural Network. In case anybody randomly inputs an wrong username and password, the output from the Fuzzy ARTMAP Neural Network is not equal to the existing hashed password. This in other words is that this user will never be able to log into the network or give access to it.

The following Table compares the training times of the MLP, RBF and the Fuzzy ARTMAP Neural Network. This shown in figure 5.

Table 1. The comparison of the network output error, training and response time of various Methods

Method	Training time	Response time	Error
Verification table	-	< 1 sec.	0
MLP network	< 60 min.	< 1 sec.	< 3%
RBF network	< 2 sec.	< 1 sec.	**0**
Fuzzy ARTMAP	< 1 Sec	< 1 Sec	0

6.2 Security performance of the proposed system

In this authentication system, the username and password is encrypted according to the modified MD5 [7]. The security of the system is based on the difficulty of the patterns to the original username and password. Thus an intruder cannot easily extract a password from the pattern even if the intruder knows the weights of the trained fuzzy ARTMAP neural Network. Therefore, only the registered user would know both the correct username and the corresponding password.

The advantage in the proposed authentication system is that an intruder may get the weights of the trained neural network and try all possible passwords to verify them until a match occurs. Such an attack is called a guess attack as described in [8] and it has become a major threat to the security of some systems especially for the users who usually tend to select an easy-to-remember password as described in [9]. These problems are overcome by the modified hashing technique in the proposed system as indicated in [7]. When there is a use of verification tables, an intruder can easily append a forged pair of username and password into the table. The proposed scheme is void of such threat as the user's name and password is combined in the trained Fuzzy ARTMAP neural network and adding a forged entry is not possible at all. In this case the entire list of usernames and passwords have to be retrained and knowing the neural network is also important.

7.Conclusions and Future Enhancement

Many authentication systems traditionally use a password table or verification table. The authentication system under discussion in this paper employs a Fuzzy ARTMAP neural network to recall the same relationship between the username and password. And this in turn replaces the traditional method of maintaining tables. The advantage of the system are:

1. Guess attack is overcome.
2. Intrusion is not permitted.
3. Forging of username and password is not possible.
4. It involves simple computations only.
5. The training time in the Fuzzy ARTMAP neural network is very short in comparison to other methods shown in figure 5.

6. This System can used to store the following securely:
    a. Passwords
    b. User Profiles
    c. Device Profiles
    d. Access Controls

In the wireless networks. It is also observed that the learning process in the Fuzzy ARTMAP is a lot faster. It is also capable of incremental stable learning. The result obtained may further be improved.

References

1. S. Bleha, M.S. Obaidat " Dimensionality reduction and feature extraction applications in identifying Computer Applications. 1991
2. V. Goyala, V. Kumara, M Singha, A. Abraham and S. Sanyalc, " A New Protocol to Counter online dictionary attacks", Computer Security, 2006.
3. R. Morris, K. Thompson, " Password Security, a case History", Communications of the ACM, Nov22(11), pp 594-597, 1979.
4. D. V. Klein, Foiling the Cracker: A Survey of and improvements to password Security", Proceedings of the Second USENIX UNIX Security workshop, pp 5-14, 1990.
5. Shahbaz Zahr Reyhani and Mehregan Mahdawi, User Authentication using Neural Network in Smart Home Networks".
6. I C Lin, H H Ou and M S Hwang, " A User Authentication system using back-propagation network", Neural Computer and Applications, 14, pp 243-249, 2005.
7. Dr. A. K. Santra and Nagarajan S, " A modified MD5 algorithm", International Journal of Network Security, August 2011 issue Vol 1. No 1
8. U Manber, " A Simple Scheme to make passwords based on the one-way function much harder to crack", Computer Security 15(2), pp 171-176, 1996.
9. D. L. Jobush, A. E. Oldehoeft, " A survey of passwoed mechanisms: Weakness and potential improvements" Security, 8, pp 587-604, 1989.
10. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer, Vision, Graphics and Image Processing*, 37, 1987, 54-115.
11. G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: fast stable learning and categorisation of analog patterns by an adaptive resonance system," *Neural Networks*, 4:6, 759-771, 1991.
12. G.A. Carpenter, S. Grossberg, and J. H. Reynolds, "ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, 4, 565-588, 1991.
13. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, 3:5, 698-713, 1992.
14. Eric Granger, Philippe Henniges, Robert Sabourin, Luiz S. Oliveira "Supervised Learning of Fuzzy ARTMAP Neural Networks Through Particle Swarm Optimization", Journal of Pattern Recognition Research 1 (2007) 27-60
15. M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, 111-147, 1974.
16. J.-F. Connolly, E. Granger, and R. Sabourin, "An adaptive classification system for video-based face recognition," Information Sciences , 2010.
17. G. Valentini, "Ensemble methods based on bias-variance analysis," Ph.D. dissertation, PhD thesis, University of Genova, 2003.
18. A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "Evaluating the performance of DNPSO in dynamic environments," in IEEE Int'l Conference on Systems, Man, and Cybernetics, Singapore, Oct. 2008, pp. 12–15.
19. W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," Information Science, vol. 178, no. 15, pp. 3096–3109, 2008.
20. X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in Genetic And Evolutionary Computation Conference, Seattle, USA, Jul. 2006, pp. 51–58.
21. E. Ozcan and M. Yy´lmaz, "Particle swarms for multimodal optimization," in Adaptive and Natural Computing Algorithms, Warsaw, Poland, Apr. 2007, pp. 366–375.
22. T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in Applications of Evolutionary Computing, Coimbra, Portugal, Apr. 2004, pp. 489–500.
23. A. Carlisle and G. Dozier, "Tracking changing extrema with adaptive particle swarm optimizer," in World Automation Congress, Orlando, Florida USA, Jun. 2002, pp. 265–270.
24. X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in IEEE Congress on Evolutionary Computation, vol. 2, Honolulu, USA, May 2002, pp. 1666– 1670.
25. H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in Applications of Evolutionary Computing, Valencia, Spain, Apr. 2007, pp. 637–

646.
26. Jean-Francois Connolly, Eric Granger and Robert Sabourin, "An Adaptive Ensemble of Fuzzy ARTMAP Neural Networks for Video-Based Face Classification"

Authors Profile

Dr. A.K.Santra is presently working as the Director (Computer Applications), at the Bannari Amman Institute of Technology in Sathyamangalam. He has close to 40 years of experience both in the industry and Teaching. He published 30 papers in various International Journals and conferences. He is presently guiding a number of students for their Ph. D. degrees. He is on the board and a reviewer in various International Journals.

Mr. Nagarajan S is presently working as Professor and Head, Associate Dean (Acaademics) at The Oxford College of Science, Bangalore. He is also a Research Scholar at Bharathiar University at Coimbatore. He has nearly about 13 years of Industry and teaching experience. He has published five international paper in International Journals and 5 in various conferences.