Intrusion Detection in Wireless Networks using FUZZY Neural Networks and Dynamic Context-Aware Role based Access Control Security (DCARBAC)

A.K. Santra
Professor and Dean,
MCA Department,
CARE School of Computer Applications, Thiruchirapalli,
Thamil Nadu

Nagarajan S.
Research Scholar,
Bharathiyar University,
Coimbatore and
Professor and Head,
The Oxford College of Science,
Bangalore, Karnataka.

Jinesh V.N.
Assistant Professor ,
The Oxford College of Science,
Bangalore, Karnataka.

ABSTRACT
This paper Proposes a Dynamic Context Aware Role Based Access Control Security Service Which Provides Multi-Level Authentications and Authorization. This model help the building of Secure Devices efficiently .This security service in a Dynamic Environment uses Fuzzy Logic theory And the DCARBAC to change the Privileges. The System uses the Attribute Values in the DCARBAC to Calculate the Type of Role which in turn is associated with the level of Permission And Security. This helps in Extending the present functionality. This Model's aim is to improve the flexibility of the Security Services in a Heterogeneous Wireless Network.

Key words: Context Aware Security, Dcarbac, Fuzzy Logic, Fuzzy Artmap.

1. INTRODNCTION
Ubiquitous technologies have penetrated in every walk of life ,which helps in spanning the Universe itself . There are a few Ubiquitous technologies like Mobile Phones , PDA'S, RFID, wireless Sensor Networks ,Wireless Network etc, to name a few. During data transmission in these environments, It is open to attacks , data alteration, intrusion , data forgery and impersonations. In order to overcome such shortcomings an improved security service is required. In recent times many networks have come into existence. This is an indicator for diverse properties and these changes dynamically according to the given environment. As of now these have been security models that have been proposed up to this proposal which have included static security functions and policies that cannot provide adaptive security services for a changing environment. Hence, the present proposal for a Dynamic Context Aware Role Based Access Control Security Model which can provide Security Services based on users and network environmental changes using Fuzzy Logic and the DCARBAC for the Dynamic Context Aware Security Services.

The structure of this paper is:
Sec2.Related work, Sec3 onwards discusses a Security model using fuzzy logic and DCARBAC and Sec10 Conclusion with Suggestion for future work.

2. RELATED WORK
Context aware computing is a new computer paradigm that determines and utilizes certain context information, such as time and location. This paradigm can provide services which the user wants if the user's context matches context in the context aware technology. In Dey's definition [1][2], context is divided into user context, computer system context, and non-classification context. This will be used in the development of context aware system according to the user's preferences.

The security model calculates the output value of DCARBAC using contextual information. Fuzzy adopts the output value of DCARBAC as a utility value and uses to calculate the security level. The security model selects a suitable authentication module using the security level

2.1 Dynamic Context Aware Computing:
It has been proved that Dynamic Role Based Access Control can manage Access Control and security, more and more mobile devices are incorporating this feature. Pervasive communication technology is becoming a everyday feature and it is changing the way of communicating with the external world. This type of DCARBAC requires the following

Tables: 1. User Location
Table 2. User Role
Table 3. Role –Permission Table and
Table 4. Mutual Exclusive role table.

Each time anybody accesses the system the first three tables are searched. Further, there is a very complex mapping of Location, users, roles and permissions. It has been observed that frequently searching the tables reduces the efficiency of access control.

Disadvantages of wireless devices are that they have less power, storage, computing and transmission abilities. Hence, performing access control in wireless environments is actually more complex than that of wired environments. There fore, any approach to access control must be relatively simple and very efficient. This paper addresses the following points:

It gives an access control algorithm and storage is reduced using the EAR decomposition and is retrieved accordingly. It also uses an ANN to train the system so that this procedure is

learnt by the system, rather than searching the tables. This algorithm assigns the user with different permissions in different sessions depending on the context aware data available at that point of time. This reduces the data storage and transmission for using only the bits making it very much easy to complement in networks where the bandwidth of the network is very low.

The anytime, anywhere access infrastructures is to enable a new generation of applications that can leverage continuously manage, adapt and finally optimization is required. The major challenge faced in Wireless applications is managing the security of the system using Access Control Lists. ACL's is a very common mechanism used in Access Control. It has been observed that the ACL's are used to check for permission to access resources or services. Another point to be noted at this juncture is such type of approach is very inadequate for wireless applications, since most proposed models do not take care of context information into consideration. There is a need for giving control in a dynamic way as the context changes according to location, time, system resources, and network security configuration etc., Therefore, access control mechanism that changes the permission of a user dynamically based on context information is very much essential. In this direction [3] have proposed a GRBAC Model and representing the system using State Machines. Using this model, it is representing the information for the new algorithm proposed and shows how it can be stored and retrieved. Then finally, show how this can be used to train the system without accessing the matrix.[19]

2.2 A procedure for security level determination

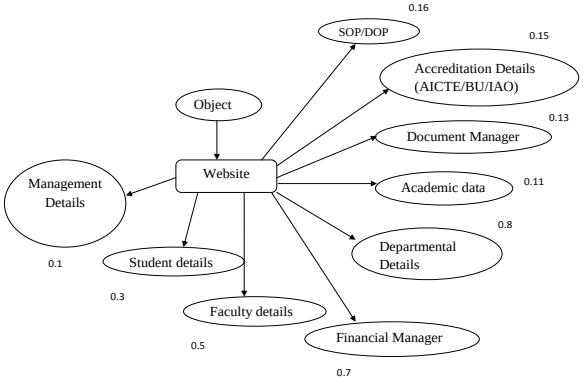

Figure:1 a

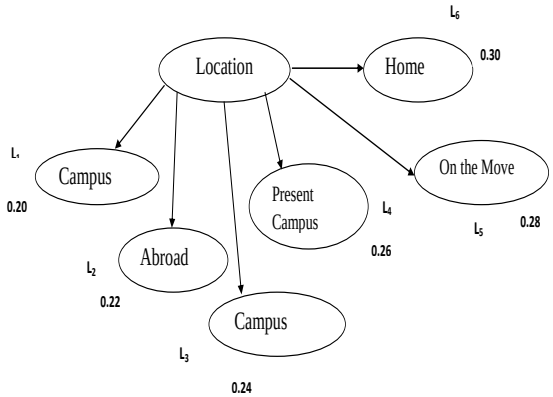

Figure:1 b

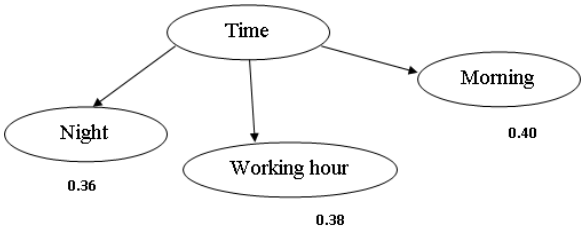

Figure:1 c

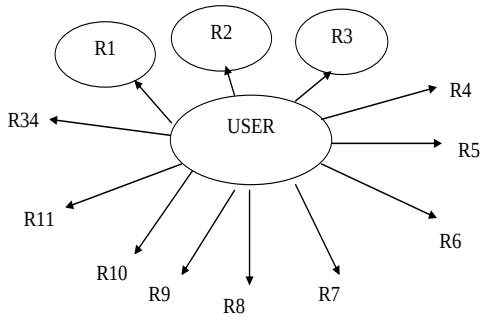

Figure:1 d

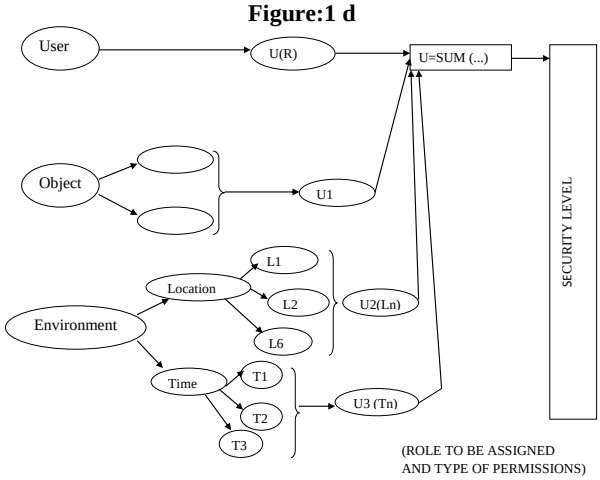

Figure:1 e

The figures above depict the components involved in the system. First, objects accessible by the user is depicted in (a). Second the locations for which the user will be given access is depicted in (b). Third the time of access Considered is depicted in (c). Fourth users associated with various roles which in tern is associated with permissions is depicted in (d). Fifth the relationship between users, object and the associated location is depicted in (e). In (e) the various objects accessible is as in (a).

3. FUZZY LOGIC

The trapezoidal distribution is defined by the minimum, lover mode, upper, and maximum parameters. The generalized trapezoidal distribution adds three more parameters: the growth rate decay rate and boundary ratio parameters. Van Dorp and colleagues formally describe the generalized trapezoidal distribution, representing the minimum lower mode, uppermode, maximum, growth rate, decay rate and boundary ratio with parameters a, b, c, d, m, n and α is given by

$$Fx(x|\theta)=C(\Theta)X \begin{cases} \alpha\left(\frac{X-\alpha}{b-\alpha}\right)^{m-1} & \text{for } a\le x<b \\ (1-\alpha)\left(\frac{X-\alpha}{b-\alpha}\right)+\alpha & \text{for } b\le x<c \\ \left(\frac{d-x}{d-c}\right)^{n-1} & \text{for } c\le x<d \end{cases}$$

With normalizing constant C(Θ) defined as

$$C(\Theta)=\frac{2mn}{2\alpha(b-a)n+(\alpha+1)(c-b)mn+2(d-c)m}$$

And where the parameter vector Θ={a, b, c, d, m, n, α}, a≤b≤c≤d, and m,n,α>0.

The trapezoid package provides functions for the probability density function(dtrapezoid), cumulative distribution function(ptrapezoid), quantile function(qtrapezoid), and random generation(rtrapezoid). The parameters a,b,c,d,m,n and α are specified by the arguments min,mode1, mode2, max,n1, n3, and alpha respectively. The argument names were chosen to avoid conflicts with names that commonly have specific meaning in R functions, such as c and n.

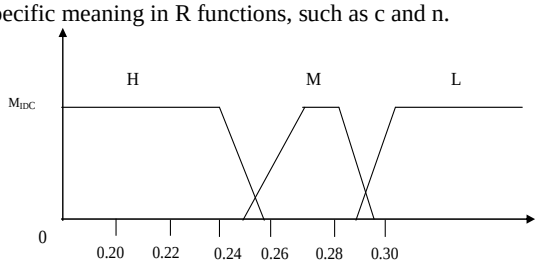

Figure:2 Trapezoidal Member function of Location

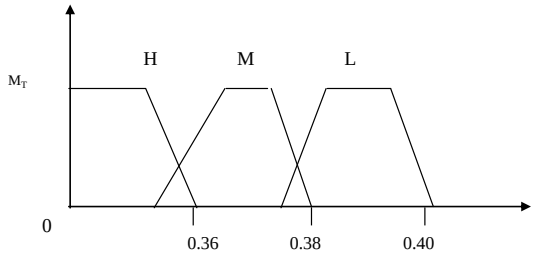

Figure:3 Trapezoidal Member function of Time

Membership function decision
F_n={H, M L}
M_{Loc}=0.26 and T=T_1
F_n={0, 0, 0, 0,1, 0}
Decision using the centroid method
FD=R_5

4. DCARBAC

The DRBAC definitions are taken from the RBAC formalisms presented in [3] and [4]USER: A user is an entity whose access is being controlled. USERS represents a set of users. ROLES: A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. ROLES represent a set of roles.

PERMS: Permission is an approval to access one or more RBAC protected resources. PERMS represents a set of permissions. LOCATIONS: Locations is the set of points from where the user accesses the resources. LOCATIONS is the set of points of access.

TIMES: Times is the time at which the user access the resources. Times is the set of time at which the user has the access.

SESSIONS: A session is a set of interactions between

subjects and objects. A user is assigned a set of roles during each session. The active role will be changes dynamically among the assigned roles for each interaction.

SESSIONS represents a set of sessions.

UA :UA is the mapping that assigns a role to a user. In the session, each user is assigned a set of roles, the context information is used to decide which role is active. The user will access the resource with the active role.

PA :PA is the mapping that assigns permissions to a role. Every role that has a privilege to access the resource is assigned a set of permissions, and the context information is used to decide which permission is active for that role.

Definition of the Agent: A Central Authority checks for the user's access rights and gives the privileges that are active for him in that session.[19]

5. EXPLANATION OF THE DRBAC MODEL

The environment considered is an educational institute. The designations are Professor, Associate Professor, Assistant Professor and Teaching Assistant. At office they will have both read and write permissions. For this we represent the locations, roles and Time in the following way:

Locations
L1 = Campuses Abroad
L2 = Campuses coming under the home country
L3 = Campuses in each City
L4 = Campuses within the city
L5 = Residence

Time
T1 = 8:00 AM to 8:00 PM (Office Hours)
T2 = 5:30 AM to 7:59 AM (Morning)
T3 = 8:01 PM to 5:29 AM (Night)
Roles
For Time T1
R1 = Professor
R2 = Associate Professor
R3 = Assistant Professor
R4 = Teaching Assistant
R5 = Professor Remote
R6 = Associate Professor Remote
R7 = Assistant Professor Remote
R8 = Teaching Assistant Remote
For Time T2
R9 = Professor
R10 = Associate Professor
R11 = Assistant Professor

R12 = Teaching Assistant
R13 = Professor Remote
R14 = Associate Professor Remote
R15 = Assistant Professor Remote
R16 = Teaching Assistant Remote
For Time T3
R17 = Professor
R18 = Associate Professor
R19 = Assistant Professor
R20 = Teaching Assistant
R21 = Professor Remote
R22 = Associate Professor Remote
R23 = Assistant Professor Remote
R24 = Teaching Assistant Remote
Permission
P1 = Append
P2 = Create.
P3 = Execute.
P4 = Get attribute.
P5 = I/O Control.
P6 = Link.
P7 = Lock.
P8 = Read.
P9 = Rename.
P10 = Unlink.
P11 == Write.[19]

6. THE FUZZY ARTMAP NEURAL NETWORK

This paper uses the Fuzzy ARTMAP Neural Network. A supervised learning algorithm is used to train the fuzzy ARTMAP feed forward Neural Network.

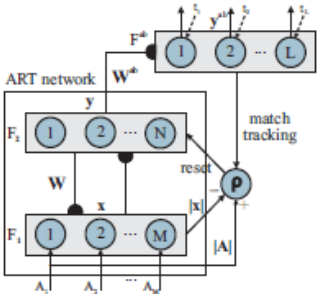

Figure:4 An Fuzzy ARTMAP neural network architecture specialized for pattern classification [4].

6.1 Supervised Training of Fuzzy ARTMAP The Fuzzy ARTMAP Neural Network

The ARTMAP Adaptive Response theory is also referred to as ART is based on the Neural Network Architecture[5], which is capable of fast, stable, on-line, unsupervised or supervised, incremental learning, classification, and prediction [6,7]. The simplified version shown in Figure 4. This is obtained by combining an ART unsupervised neural network [6] to its map field. By employing fuzzy ART as the ART network which is capable of processing both analog and binary-valued input patterns, such an ARTMAP architecture called fuzzy ARTMAP [5].

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_2layer connections. Each F_2node *j* represents a recognition category that learns a prototype vector $\boldsymbol{w}_J^{ab}.=(\boldsymbol{w}_{J1}^{ab}.\ \boldsymbol{w}_{J2}^{ab}.\ ....\boldsymbol{w}_{JL}^{ab})$. The F^2layer of fuzzy ART is connected, through learned associative links, to an *L* node map field ^{Fab}, 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 $W_i^{ab}=(W_{i1}^{ab}\ W_{i2}^{ab}\ .....\ W_{jL}^{ab})$ links F_2node *j* to one of the *L* output classes.

The training set patterns $\boldsymbol{a}=\{a_1,a_2,\ldots,a_m)$and their corresponding binary supervision patterns $\boldsymbol{t}=\{t_1,t_2,\ldots,t_L)$ part of the ARTMAP classifiers learn an arbitrary mapping between the training set pattern and binary supervision patterns. 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.[8]

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 $\beta\in[0,1]$, the choice $\alpha>0$, the match tracking $0<\varepsilon<<1$, and the baseline vigilance $\rho\text{-}\in[0,1]$ parameters are set.

2. Input pattern coding—When a training pair $\{\mathbf{a},\mathbf{t}\}$ 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 $\mathbf{A}=\{\mathbf{a},\mathbf{a}^c)=\{a_1,a_2,\ldots,a_m;\ a^c_1,\ a^c_2,\ \ldots,a^c_m)$, where $a^c_1;=(1-a,.)$, and $a_1\in[0,1]$. 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:

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

where \.\ is the L_1norm operator defined by $|\mathbf{w}_i|=\Sigma_{i=1}^{M}\backslash w_{ti}\backslash$, $\wedge$ 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 $\mathbf{y}=\{y_l,y_2,\text{-},y_N)$ 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 $\rho\in[0,l]$:

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

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 $\{\mathbf{a},\mathbf{t})$ 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 $\mathbf{y}^{ab}=\{y_1^{ab}, y_2^{ab}.., y_L^{ab}\}=\mathbf{t} \wedge \mathbf{w}_i^{ab}$ 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$\varepsilon=0^+$, 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 $\mathbf{w}_{i:}$, 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:

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

where β is a fixed learning rate parameter. The algorithm can be set to slow learning with $0<\beta<1$, or to fast learning with $\beta=1$. With complement coding and fast learning, fuzzy ARTMAP represents category j as an m-dimensional hyper rectangle R_i 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 $\mathbf{w}_i$ records the largest and smallest component values of training subset patterns **a** assigned to category j. The vigilance test limits the growth of hyper rectangles – a ρ close to 1 yields small hyper rectangles, while a ρ close to 0 allows large hyper rectangles. 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)*. [8]

7.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, $SSFC=\Sigma_i\Sigma_j.(W_{ij}(e)-W_{ij}(e-1))^2<0.001$.

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 [9], 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 an is 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. [8]

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

where *f(h,t)* is the classification rate of FAM for a given vector of hyperparameters h = *(α,β,ε,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 [10], [11], [12], [13]. Change detection and memory adjustment mechanisms for DPSO are presented in [14], [15], [16], [17]. In this paper, the adaptive classification system (ACS) uses a Dynamical Niching PSO (DNPSO) [10] to maximize FAM classification rate as a function of its hyper parameters vector h = *(α,β,ε,p)* when learning new data blocks

D_t(Figure 1). The optimization space is defined by the four FAM hyper parameters$(\alpha\in$[0.001,100], $\beta\in$ [0,1], ε∈ [—1,1], and p ∈ [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 [10]. 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 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.[8]

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_n. 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}←$FAM^{s\wedge rt}$
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_n,(r), FAM_{est}, $f(h_n(t),t)$}
12: end if
13:end for
14: T = T + 1
15: end while
16: for each particle n, where $1 \leq n \leq N$ do
17:FAM_n^{start}←FAM_n
18: end for

updated when a new $D_t$is presented to the system, *before* the iterative DNPSO process.[8]

7.2 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^{start} 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 up date 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^{start} (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 $FAM^{st\wedge rt}$, serves as a starting state and is copied to FAM_{es}tprior 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 [18] 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. [17][8]

8. DYNAMIC CONTEXT AWARE ROLE BASED ACCESS CONTROL SECURITY MODEL.

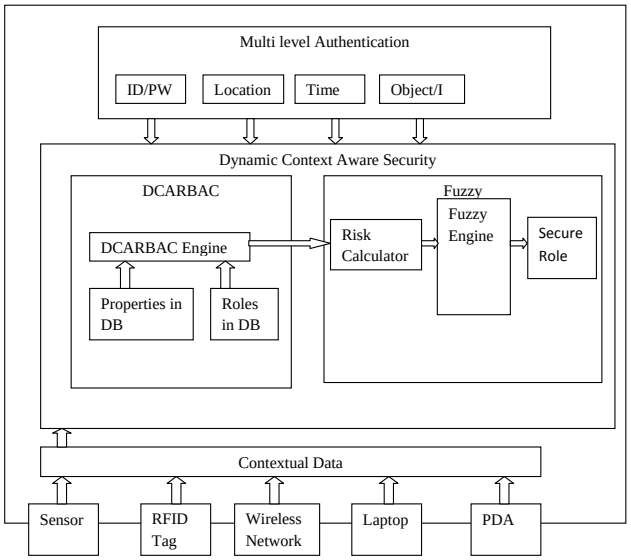

Figure:9 Dynamic Context Aware Security Model

Architecture Based on fuzzy Logic and DCARBAC.The Security Model Proposed in this Paper is as follows:
U: The User Note Demands for a Transaction.
L: The User's Location.
OP: Read Write……..
The Context stated alone is handled in the ACS (access Control Server) which is capable of efficient authorization .i.e ., Every Service Transaction of Authorization Is Accomplished through a QuadrupleT= <U, L, RO, OP>, this means that the user does that operation on the object in a certain Location , DCARBAC usesan algorithm to evaluate the T.[20]

8.1 DCARBAC For Adaptive Security

Whenever a user logins in with a User Name and Password for a Service. This Model will use the Fuzzy Logic to find the information of the user, Object, Environment and types of Operations. The Fuzzy Trapezoidal Function decides the role and decides on the level of permissions associated. If It is expressed that U(X)=0 And U(X*)=1 as the lowest grade of security then the evaluation is to associate the permissions with the role, which is decided as per the attributes received pertaining to the Username, Password, Location Time and Resource. The function Can be defined as U(Un, Pn)=∑nu=0 Ui(Ri),… ……. (1)
Requirement 1
The Username, Password, Environment.
Requirement 2
Associated co-efficient for each attribute.
Requirement 3
A risk calculator is needed to mitigate the risk involved with the user .

Algorithm 1: In order to satisfy the requirement 1 attribute value is added to the object and environment information in DCARBAC.

Algorithm 2: In the case of a Role determination, the administrator's location and access time are very important in deciding the security level than those of the employees.
Since the administrators needs to have rights to have transactions whenever and wherever it is required. Thus, there is an introduction of values to attributes of objects Environment and user name and passwords in order to satisfy the requirement 2.

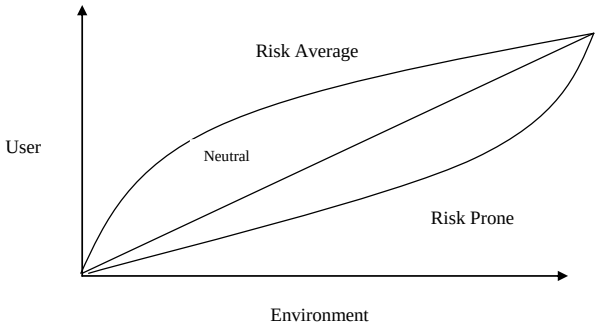

Figure:6 The Three types of Risk Preferences.

Algorithm 3: The risk averse means that from the system's view point the user cannot be provided with a low security authentication method, while taking a high risk and risk prone means that the user is provided with lower authentication than others. Risk neutral is the mixed method between both of the two like the current system.
The administrator of the security model in the applied system has the flexibility to fix the risk-preference of the user in accordance with the user environment.

The Set Context Value Algorithm

```
// Algorithm 1
Set Context Value (user, location, Role)
Role=Get value of DCARBAC (user, location);
//Algorithm2
Location=Get value of DCARBAC (Location);
Time=Get value of DCARBAC (Time);
//Algorithm3
User=Get value of DCARBAC (Role)
Return Context value;
End
Get Value Of DCARDAC (Parament Method )
Switch (parameter)
{
// Algorithem 1
Case Role.
Get Permission From Role in DCARBAC
Else If Method Is Time Then
Get This Information from time in DCARBAC
// Algorithem 3
Case Role
If Method is RiskprovenThen
Get This Information from Riskproven Note in DCARBAC
Else If method isuriskNeutral Then
Get This Information from uriskNeutral Role In DCARBAC
Else if Method isUrisk averse Then
```

Get This Information From URiskAverse Role In DCARBAC.
Default :,
}
// END of Switch
Return Value:
// Value is Return value from each Role in DCARBAC
End;
// End of Getvalue Of DCARBAC. [20]

The adaptive Security Algorithm.

The User's Security is determined in this proposal using Fuzzy Trapezoidal function to decide on the level of Authentication in the extended DCARBAC. This uses a Security Method based on the Fuzzy Trapezoidal Algorithm as defined in [ ].Variables that are used as inputs in this adaptive security algorithm consists of User Password, Time , location which is got at the instance of time from the Context in the CARBAC. This is given to the Fuzzy Trapezoidal Algorithm Model. This computes the role to be assigned and in Communication with the policy manager decides on the level of Authentication and Permissions for access. Table 2 Shows the adaptive Algorithm. Security Level (Role, Risk Preference)

// SL : Determining The Security Level .
SL: Fuzzy Trapezoidal (ROLE , Risk Preference).
Return ;
Fuzzy Trapezoidal (Role, Risk Preference).
//Determines the Security Level based on the values received at that Instance.
//Rp , L,T, OB , P Is to be Determined. L by the System Administrator and will change as required.
Decide risk preference according to context values got for theRp, L, T, ob, P from DCARBAC and decide using the policy manager.
Level of Access
Else
Deny Access
End
Return value accordingly.
Get Risk (Rp).
//Determine Risk level as follows:
//Risk Prone: user is risk prone
//Risk Neutral: user is risk Neutral
//Risk Averse: user is risk averse
Calculate the Level according to risk tendency
If Rp is a Risk prone then
Return Rp= -1;
Else if Rp is Risk Neutral then
Return Rp=0;
Else if Rp is Risk Averse then
Return Rp=1;
End:The Learning Module [20]

9. THE EVALUATION OF ADAPTIVE SECURITY LEVEL

The performance of the proposed adaptive security algorithm goes through a comparison of the security level of the Faculty, administrator and students as decided by the Risk-preference under similar conditions.First, the security level as decided by Risk preference under similar conditions is shown in the figure. It is a measure of data according to the time element and location, which is sensitive to domestic and Foreign Locations advantages the proposed of security Model.

The following are the advantages.

1. It provides multiple authentications according to Context.

E.g.1.If user is accessing through wireless network then apart from his Username/password location and time is taken to determine level of Security.

E.g.2. Through RFID tag along with UN/PWD,the RFID number is considered. Etc...

2. Authorization through the DCARBAC.
3. It can operate in a heterogeneous environment.
4. API is provided for efficient development.

10. CONCLUSION AND FUTURE ENHANCEMENT

The paper suggests a dynamic Context Aware Role Based Access Control Service in a Context Aware environment. The proposed model has multiple authentications, fuzzy Trapezoidal algorithm and the DCARBAC .It will help user to securely access Context aware application.

The fuzzy Trapezoidal algorithm helps to decide the level of security as the location changes.In the process the DCARBAC is used in the Authentication process. Therefore, in the proposed module the data can be accessed securely.

In the future this model can be extended to make the system to learn to dynamically the entire process to automate it and avoid the interference of the administrator to grant the Risk preference levels. Tools to reduce overload of factors affecting the assignment of Roles also can be look at.

11. REFERENCES

[1] A K Dey, "Providing Architectural Support for building Context aware Applications", PhD dissertation,Georgia institute of technology,2000.

[2] Gualing Chen, "A survey of Mobile computing Research", darmouth university computer science technical report TR2000-381, 2000.

[3] Leveraging IPsec for Distributed Authorization, Trent Jaeger, David King, Kevin Butler, Jonathan McCune, Ramon Caceres, Serge Hallyn, Joy Latten, Reiner Sailer and Xiolan Zhang. nsrc.cse. psu.edu /tech _report/NAS-TR-0037-2006.pdf, 2006

[4] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in Applications of Evolutionary Computing, Coimbra, Portugal, Apr. 2004, pp. 489–500.

[5] 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.

[6] 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.

[7] 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.

[8] 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

[9] M. Stone,"Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, 111-147, 1974.

[10] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "Evaluating the performance of DNPSOin dynamic environments," in IEEE Int'l Conference on Systems, Man, and Cybernetics, Singapore, Oct. 2008, pp. 12–15.

[11] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," Information Science, vol. 178, no. 15, pp. 3096–3109, 2008.

[12] 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.

[13] E. Ozcan and M. Yy´lmaz, "Particle swarms for multimodal optimization," in Adaptive and Natural Computing Algorithms, Warsaw, Poland, Apr. 2007, pp. 366–375.

[14] 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.

[15] 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.

[16] 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.

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

[18] G. Valentini, "Ensemble methods based on bias-variance analysis," Ph.D. dissertation, PhD thesis, University of Genova, 2003.

[19] Dr. A K Santra and Nagarajan S, A Method of Access in a Dynamic Context aware Role based Access Control Model for wireless Network, International Jurnal of Computer and Network security, Vol.2, NO.4,April 2010.

[20] Kiyeal Lee, Seo Khwan Yang, Sungik Jun and Mokdong Chung, "Context-Aware Security Service in RFID/USN Environments using MAUT and Extended GRBAC ", Journal of Information Assurance and Security 2 (2007) 250-256.