A Hybrid GA-PSO Fuzzy System for User Identification on Smart Phones

Muhammad Shahzad, Saira Zahid, Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer and Emerging Science (FAST–NUCES)
Islamabad 44000, Pakistan
{muhammad.shahzad, saira.zahid, muddassar.farooq}@nexginrc.org

ABSTRACT

The major contribution of this paper is a hybrid GA-PSO fuzzy user identification system, UGuard, for smart phones. Our system gets 3 phone usage features as input to identify a user or an imposter. We show that these phone usage features for different users are diffused; therefore, we justify the need of a front end fuzzy classifier for them. We further show that the fuzzy classifier must be optimized using a back end online dynamic optimizer. The dynamic optimizer is a hybrid of Particle Swarm Optimizer (PSO) and Genetic Algorithm (GA). We have collected phone usage data of 10 real users having Symbian smart phones for 8 days. We evaluate our UGuard system on this dataset. The results of our experiments show that UGuard provides on the average an error rate of 2% or less. We also compared our system with four classical classifiers – Naïve Bayes, Back Propagation Neural Networks, J48 Decision Tree, and Fuzzy System – and three evolutionary schemes – fuzzy system optimized by ACO, PSO, and GA. To the best of our knowledge, the current work is the first system that has achieved such a small error rate. Moreover, the system is simple and efficient; therefore, it can be deployed on real world smart phones.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Authentication*

General Terms

Algorithms, Design, Security

Keywords

Hybrid GA-PSO-Fuzzy, Authentication, Chromagent

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

1. INTRODUCTION

Genetic algorithms are finding their applications in a number of emerging real-world applications: biomedical informatics, computational finance, computer graphics and games, and network/host security. However, their utility in information security systems has received little attention. Specifically their usefulness in developing intelligent user authentication schemes to ensure legitimate access to a device's resources has not been explored. Moreover, their use in the emerging smart phones market is absolutely non-existent. We believe that genetic algorithms, if developed with an engineering vision, can play a vital role in targeting intelligent security solutions for next generation mobile devices.

A recent study of mobile phone users in UK, US and Japan by a security analyst of McAfee reports that 58% of the mobile phone users are seriously worried about the data and services security of their mobile phones. The survey in [1] and [5] also report similar trends. According to these surveys, the information security on mobile phones is a serious concern of both users and manufacturers. The mobile operators mostly use Personal Identification Number (PIN) for identifying their legitimate users. But this scheme has a number of serious flaws: (1) the authentication schemes based on passwords can be easily broken by stealing or overhearing the password, and (2) the users also do not feel comfortable to enter the passwords every time they want to use the mobile phones [4].

In this paper, we show that a better and robust approach to identify a legitimate user of a mobile phone is to learn his/her behavior of using the mobile phone. We set four objectives for our UGuard identification system: (1) correctly identify an imposter and the legitimate user, (2) user's identification within 8 calls, (3) if an imposter is detected, block the mobile phone to ensure information security, and (4) the system should be simple and efficient to facilitate its deployment on smart phones.

We, therefore, propose our UGuard system which learns the behavior of the legitimate user and blocks any imposter based on a fuzzy classifier that optimally maps the diffused features space of a mobile phone user to his/her profile. It also utilizes an online dynamic optimizer that is a hybrid of (PSO) and (GA) at the back end for continuous evolution of the fuzzy system. This caters for varying usage pattern of a user. PSO and GA are well-known for providing efficient, online solutions to dynamic and time-varying optimization problems. The results of our experiments justify the use of the hybrid approach. We compare our system with a number of bio-inspired and classical machine learning classifiers

on a real dataset of 10 mobile phone users collected over a period of 8 days. (To the best of our knowledge, no such dataset is currently available.) Our system significantly outperforms other approaches and achieves an error rate of less than 2%. Consequently, the system provides significantly better accuracy not only in detecting an imposter but also the legitimate users. Once we detect an imposter, we can report it to the cybercrime division of Police to track and ultimately catch the culprit. We believe that such a system has the potential to become an integral part of the operating systems of smart phones.

The rest of the paper is organized as follows. We explain the architecture of our UGuard system in Section 2. In Section 3, we report the results of our comparative study and Section 4 refers to related work. Finally, we conclude our paper with an outlook to our future research.

2. ARCHITECTURE OF UGUARD

In this section we give a detailed description of our proposed system called "UGuard". The aim of our work is to design a system for smart phones that should detect the legitimate user from an imposter. An imposter should always be detected as an imposter without any error. This error is quantitatively represented by False Acceptance Rate (FAR) and should ideally be 0%. Similarly it is important that the legitimate user should always be granted access without any error. This error is called False Rejection Rate (FRR). Ideally FRR should also be 0% because a large FRR will quickly become frustrating for the owner of the smart phone. Last but not least, the detection of an imposter must be done in the smallest possible time.

We now discuss the architecture of UGuard. Figure 1 shows the block diagram of the complete system. First of all the features are extracted from the log of a user and k-means clustering is used to generate an initial rule base for the fuzzy inference system. The rule base of the fuzzy system is optimized online in realtime with the help of a hybrid PSO-GA scheme. The system first runs in the training mode and then is deployed for classification of users in realtime. Algorithm 1 describes the functioning of our system.

2.1 Feature Extraction

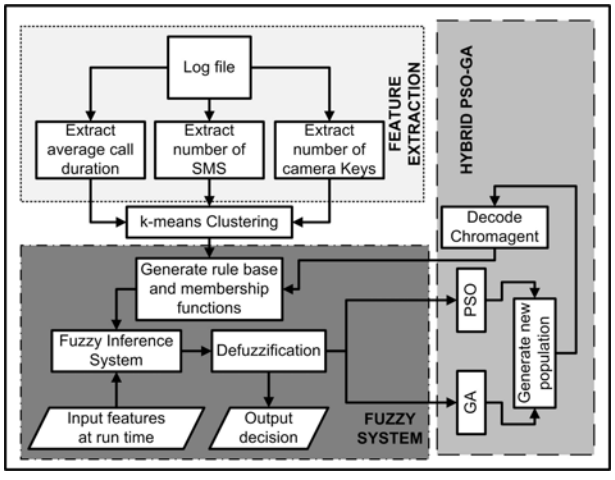

Figure 1: Top Level Diagram of UGuard

We require a feature set that can be utilized to distinguish an imposter from a legitimate user. To this end, we use a set of three features. It is important to mention here that we divide the log of a user into profiles. Each profile consists of 8 dialed calls. (This number will be shortly justified during the discussion of the experiments). The features extracted from each profile are listed below:

Average Call Duration. We take the average call duration of 8 calls in a profile.

Number of SMS. This is the total number of short messages that are sent in a single profile.

Camera Usage. The total number of camera key hits in one profile.

Table 1 lists data of 10 real-world users, which include young students, professional corporate executives, senior citizens and software engineers. The table shows the number of profiles of each user along with the average values of the three features used for each user.

Algorithm 1 *Main*

procedure Main(UserProfile,TrainingProfile1,TrainingProfile2)
{Profiles contain Call Times, Number of SMSs and Camera Keys}
training data ⇐ extract features from UserProfile and TrainingProfiles
$PopSize \Leftarrow 30$ {total number of chromagents}
$EliteCount \Leftarrow 2$ {number of elite chromagents selected as it is}
$XoverFrac \Leftarrow 0.6$ {fraction of chromagents selected for cross over}
$TotalIter \Leftarrow 50$ {total number of generations}
$StallIter \Leftarrow 25$ {number of iterations during which if the fitness does not improve, the algorithm stops}
$PersonalInc \Leftarrow 0.35$
$GlobalInc \Leftarrow 0.40$
$CommunicationInterval \Leftarrow 5$ {generations after which GA and PSO communicate}
$OptimumChromagent \Leftarrow$ Training($PopSize, EliteCount, XoverFrac, TotalIter, StallIter, PersonalInc, GlobalInc, CommunicationInterval$,training data)
while 1 **do**
$TestingProfile \Leftarrow$ extract features from real time user data
return FuzzyInference($OptimumChromagent$,$TestingProfile$)
end while
end procedure

Table 1: Feature table of users

Users	Total profiles	Average call duration(secs)	Average SMS	Average Camera keys
x1	11	24.1	37.2	16.4
x2	15	11.3	17.1	8.22
x3	10	154.1	8.86	2.82
x4	21	32.2	15.3	7.45
x5	9	82.7	7.21	18.9
x6	14	11.4	0.21	1.69
x7	13	1452.3	0.83	1.21
x8	22	171.4	35.4	13.37
x9	12	84.2	17.6	0.51
x10	8	26.8	52.7	8.3

Figure 2 shows the plot of the mean values of the three features for all profiles of each user. It can be seen in the figure that the features of various users are quite fused into each other and classification of such diffused datasets is a daunting task for classical machine learning algorithms. Therefore, we propose an online real-time fuzzy system for classification whose rule base is optimized using a hybrid PSO-GA scheme.

2.2 Fuzzy Classifier

We are working on a two-class classification problem as we need to distinguish between a legitimate user and an

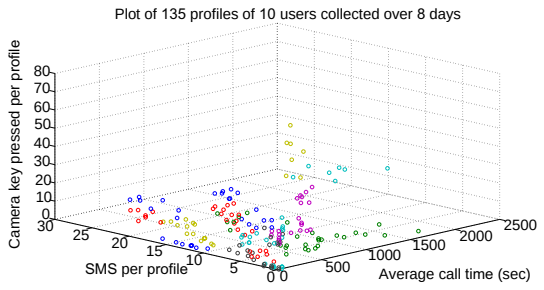

Figure 2: Diffused nature of the features used for classification

imposter. It is obvious in Figure 2 that we cannot use standard classifiers because it is not possible to assign crisp class labels to the diffused set. Consequently, it is justified to deploy a fuzzy classifier because: (1) it assigns a given data point a degree of membership to all available classes, and (2) it uses linguistic variables that provide traceable and interpretable steps and logic statements, which ultimately leads to the class prediction for a given data point [12]. In a fuzzy system, we also define its inputs or outputs in terms of linguistic fuzzy variables, which represent rules and facts. A fuzzy system also uses a database and a rule base. Algorithm 3 shows how our fuzzy inference system works. We now elaborate how we design database and rule base for our fuzzy system.

Recall that we give a set of three features as an input to the system and it classifies a given user as a legitimate user or an imposter (2 outputs). We provide 3 features mentioned in Section 2.1 to our fuzzy system. The system gives a positive output for a legitimate user and a negative output for an imposter. We then define a certain range for each variable after observing its variation in the dataset; however, the output variables vary in the interval [0,1]. A fuzzy variable can have a number of partitions, each having its own membership function. We consider six fuzzy partitions for each fuzzy variable. The sets are defined in terms of six linguistic labels defined by the set L:

$$L = \{very\ large, large, medium, moderately\ medium, small, very\ small\}$$

As the etymology suggest, the labels *very large* to *very small* represent a decreasing tendency of the corresponding variable to belong to a particular set. For simplicity our output variables have only three linguistic labels: *low*, *medium* and *high*; each label quantifies the grade of membership of an input variable to an output variable. After analyzing the patterns of features' set, we defined trapezoidal membership functions. The trapezoidal functions provide a range in which their value remains constant at 1. This property helps in mapping a user whose values vary in the range of the flat portion to model his/her normal behavior. Our experiments show that this significantly improves the overall accuracy of our system.

We use simple k-means algorithm for rule base generation by doing clustering. The clusters with the smallest number of data points are considered outliers and are discarded. We define the centroid of a cluster in the form of $(x1, x2, x3)$, where $x1$, $x2$, and $x3$ are the values of the first, second and third features respectively. We search their values in the

Algorithm 2 *Training*

procedure Training($PopSize$, $EliteCount$, $XoverFrac$, $TotalIter$, $StallIter$, $PersonalInc$, $GlobalInc$,$CommunicationInterval$, training data)
make sample chromagent s by $k-means$ clustering
$Pop[:] \Leftarrow$ GenerateInitialPopulation($PopSize, s$)
$GAPop \Leftarrow Pop[1 : \frac{\text{size}(Pop)}{3}]$ {chromagents used for GA}
$PSOPop \Leftarrow Pop[\frac{\text{size}(Pop)}{3}+1$:size$(Pop)]$ {chromagents for PSO}
for $j = 1$ to size($GApop$) **do**
  $GAPopFitness[j] \Leftarrow$ComputeFitness($GAPop[j]$,training data)
end for
for $j = 1$ to size($PSOPop$) **do**
  $PSOPopFitness[j] \Leftarrow$ComputeFitness($PSOPop[j]$,training data)
end for
$PrevBestFitness \Leftarrow$ best value in $GAPopFitness[:]$ and $PSOPopFitness[:]$
$SameFitnessIter \Leftarrow 0$
$CommunicationIter \Leftarrow 0$
$i \Leftarrow 0$
while $i \leq TotalIter$ and $SameFitnessIter \leq StallIter$ **do**
  $GAPop \Leftarrow$ GeneticAlgo($GAPop, GAPopFitness[:]$, $EliteCount, XoverFrac$)
  $PSOPop \Leftarrow$ PSOAlgo($PSOPop, PSOPopFitness[:]$, $PersonalInc, GlobalInc$, training data)
  for $j = 1$ to size($GAPop$) **do**
    $GAPopFitness[j] \Leftarrow$ ComputeFitness($GAPop[j]$, training data)
  end for
  for $j = 1$ to size($PSOPop$) **do**
    $PSOPopFitness[j] \Leftarrow$ ComputeFitness($PSOPop[j]$, training data)
  end for
  $CommunicationIter \Leftarrow CommunicationIter + 1$
  if $CommunicationIter == CommunicationInterval$ **then**
    replace worst chromagent of $GAPop[:]$ with best of $PSOPop[:]$
    replace two worst chromagents of $PSOPop[:]$ with two bests of $GAPOP[:]$
    $CommunicationIter = 0$
  end if
  $CurrentBestFitness \Leftarrow$ best value in $GAPopFitness[:]$ and $PSOPopFitness[:]$
  if $PrevBestFitness == CurrentBestFitness$ **then**
    $SameFitnessIter \Leftarrow SameFitnessIter + 1$
  else
    $SameFitnessIter \Leftarrow 0$
  end if
end while
return overall best chromagent in $GAPOP[:]$ and $PSOPop[:]$
end procedure

corresponding fuzzy sets, determine their degree of membership to the fuzzy partitions in which they lie, and choose the partition with the maximum degree of membership. For example, if we get a value for the first feature that belongs to the fuzzy partition *very small* with 0.7 and to fuzzy partition *small* with 0.3, then we choose *very small* as the condition for this fuzzy variable in the antecedent of the rule. The conditions for the other two variables are determined in a similar fashion to completely define the antecedent of the rule. To determine the consequent of a rule, we find the density of the cluster of the centroid for which we are defining an antecedent of the rule. If a cluster has *high*, *medium* or *low* density then the output belongs to the fuzzy partition *high*, *medium* or *low* respectively in the consequent of the rule. In this way we define the whole rule base using the centroid of all the clusters. We usually start with five clusters in our database. An inference system applies the rule base to the inputs in order to determine the outputs. We use standard Mamdani Inference System [7], which uses our customized defuzzification technique explained in Algorithm 3.

Algorithm 3 *Fuzzy Inference System*

procedure FuzzyInference(s, $UnknownProfile$)
generate input membership functions, output membership functions and rule base according to chromagent s
apply fuzzy inference system generated above to$UnknownProfile$
$ValuePos = 0.3 * \int_0^{x_1} A(x)dx + 0.7 * \int_{x_1}^{x_2} A(x)dx + 1 * \int_{x_2}^{1} A(x)dx$
$ValueNeg = 0.3 * \int_0^{x_1} A(x)dx + 0.7 * \int_{x_1}^{x_2} A(x)dx + 1 * \int_{x_2}^{1} A(x)dx$
{x_1 and x_2 are the intersection points of low,$medium$ and $medium$,$high$ membership functions of each of the two output membership functions. x is dummy integration variable}
if $ValuePositive > ValueNegative$ **then**
  return OWNER
else
  return IMPOSTER
end if
end procedure

Algorithm 4 *Initial Population*

procedure GenerateInitialPopulation($PopulationSize, s$){s is a sample chromagent}
for $i = 1$ to $PopulationSize$ **do**
  $population[i] \Leftarrow s$
  for $j = 1$ to 159 **do**
    $x \Leftarrow$ random number
    $y \Leftarrow$ random number
    if $x > 0.5$ **then**
      $population[i, j] \Leftarrow population[i, j] + 7 * y$
    else
      $population[i, j] \Leftarrow population[i, j] - 7 * y$
    end if
  end for
end for
return *population*
end procedure

The FAR and FRR values of our fuzzy system are not very good but our analysis reveals that they can be improved considerably if we use a dynamic optimizer that dynamically maps our fuzzy system to the diffused continuously changing features' set.

2.3 Optimizer

As mentioned above, an initial rule base is generated using k-means clustering. But we need to dynamically optimize it to cater for changing behavior of a user. Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) are well-known online optimizers for dynamic environments.

The main idea of PSO is to use a swarm of agents spread on the problem space, and these agents through local interactions tend to find an optimal solution of the problem. The feature that makes PSO successful is the communication between the agents. This, in essence, is the concept of feedback which is utilized by every agent to converge to the best location. GA, however, does not utilize feedback explicitly; rather it uses genetic operators of selection, crossover and mutation to find the best solution. In our hybrid approach we combine feedback of PSO and diversification concepts of GA.

We will call the individuals in the population/swarm as "chromagents". Both PSO and GA use the same type of chromagents. We now describe the structure of chromagents in detail. Our hybrid PSO-GA optimizer is used to evolve the following features of the rule base of the fuzzy system:
(1) Ranges of fuzzy partitions of input and output variables;
(2) Total number of rules;
(3) Conjunction type of the rules i.e. either AND or OR;
(4) Rules.
In order to bound the computational complexity of the system, we allow maximum of 20 rules. The structure of a chromagent is shown in Table 2. We use Pittsburgh encoding approach introduced in [2]. Each chromagent consists of 159 genes. The size of a chromagent = 1(representing number of rules) + 3(for each input variable)* 10(representing ranges of each fuzzy partition for a single variable) + 2(for each output variable)* 4(representing ranges of each fuzzy partition for a single variable) + 6(for each rule)*20(maximum number of possible rules)=159

We now explain the other aspects of the system.

2.3.1 Initial Population

Our system begins with a random initial population. For this purpose, we take the rule base generated by the k-means clustering and encode it accordingly in a chromagent. This chromagent serves as a sample and a total of 29 more chromagents are generated that are variants of the sample chromagent; however they differ substantially from one another to ensure diversity. In this way we get an initial population of 30 agents. One chromagent takes 273 bytes of RAM. Thus a population of 30 will take up less than 8KB of RAM which is negligible as compared to few tens of MB of RAM available in ordinary smart phones. Algorithm 4 details the process of generating the initial population.

2.3.2 Fitness Function

Once the initial population of chromagents is generated, the next important step is to evaluate each individual and assign it a fitness value. We define fitness as:

$$fitness = 1 - (0.75 * (1 - FAR) + 0.25 * (1 - FRR)) \quad (1)$$

The motivation behind assigning more weight to FAR compared with FRR is twofold: (1) if an imposter is classified as a legitimate user (FAR) it represents a serious threat to the data on the mobile phone, however, (2) if a legitimate user is classified as an imposter (FRR) it is a mere annoyance to the user without any threat to the data. The weight of 0.75 for FAR and 0.25 for FRR are empirically chosen to provide an acceptable tradeoff between FAR and FRR. The pseudo code for evaluating the fitness function is described in Algorithm 5.

Once all chromagents are assigned fitness values, the population is divided into two subgroups. The first subgroup consists of 20 chromagents that are used by PSO for optimization. The second subgroup consists of remaining 10 chromagents, which are optimized by GA. For the first time,

Table 2: The description of a chromosome

Genes	Encoding description
$g_1 - g_{10}$	fuzzy partitions of first input variable
$g_{11} - g_{20}$	fuzzy partitions of second input variable
$g_{21} - g_{30}$	fuzzy partitions of third input variable
$g_{31} - g_{34}$	fuzzy partitions of first output variable
$g_{35} - g_{38}$	fuzzy partitions of second output variable
Note that $g_i \leq g_{i+1}, \forall i = 1 - 9 \vee i = 11 - 19 \vee i = 21 - 29 \vee i = 31 - 33 \vee i = 35 - 37$	
g_{39}	represents the total number of rules
Note that $0 \leq g_{39} \leq 20$	
$g_{40} + k * 6$	represents the conjunction type
value of g_{40} is 8 or 9	
$(g_{41}, g_{42}, g_{43}) + k * 6$	represents partition values of three inputs
$(g_{44}, g_{45}) + k * 6$	represents partition values of two outputs
Note that $k = 0$ *to* $g_{39} - 1$	

Algorithm 5 *Fitness Function*

procedure ComputeFitness(s, training data){s is a chromagent}
if s is a valid chromagent **then**
  $Profile \Leftarrow$ a randomly selected known profile from training data
  $O_T \Leftarrow 0$ {number of correctly identified owner}
  $O_F \Leftarrow 0$ {number of false identified owner}
  $I_T \Leftarrow 0$ {number of correctly identified intruder}
  $I_F \Leftarrow 0$ {number of false identified intruder}
  $UserIdentity \Leftarrow$ FuzzyInference(s,$Profile$)
  if OWNER identified as OWNER **then**
    $O_T \Leftarrow O_T + 1$
  else if OWNER identified as IMPOSTER **then**
    $O_F \Leftarrow O_F + 1$
  else if IMPOSTER identified as IMPOSTER **then**
    $I_T \Leftarrow I_T + 1$
  else if IMPOSTER identified as OWNER **then**
    $I_F \Leftarrow I_F + 1$
  end if
  $FRR \Leftarrow O_F/(O_F + O_T)$
  $FAR \Leftarrow I_F/(I_F + I_T)$
  $fitness \Leftarrow 1 - (0.25 * (1 - FRR) + 0.75 * (1 - FAR))$
  return $fitness$
else
  $fitness \Leftarrow 1$ {worst fitness value is 1 and best is 0}
  return $fitness$
end if
end procedure

we randomly assign individuals to each group. We now discuss GA and PSO optimizers and the communication process between them.

2.4 Genetic Algorithm

The genetic algorithms (GA) [6] are well-known for providing acceptable solutions to dynamic optimization problems. Our GA uses a set of 10 chromagents and applies the selection, crossover and mutation operators on them. Algorithm 6 explains the working of GA in our system.

2.4.1 Next Population Generation

A certain number of chromagents (elite count) having the best fitness values are automatically selected as the chromagents of next generation. This helps in maintaining good configurations of a fuzzy system. We take 20% of the chromagents in the elite count. Our crossover fraction is 0.6 and mutation fraction is 0.2. The crossover and mutation functions are explained in Algorithm 6 and we have used standard "stochastic uniform" method for selection. It is important to note here that we have done an extensive analysis on the population size, the number of generations, selection criteria, and crossover and mutation fractions as well as types of these functions and empirically determined the optimum values for all of these features reported in this paper. For brevity the analysis has not been reported in this paper. Once selected, our proposed crossover and mutation operators are applied to chromagents and this produces new configurations of the fuzzy system. The next generation gets its required number of chromagents through elite count, selection, crossover and mutation operators. Algorithm 6 explains our crossover and mutation operators. Once we get the next generation of population, the fitness value of each chromagent in the new population is evaluated and the cycle repeats. The stopping criteria will be discussed shortly.

2.5 Particle Swarm Optimization

PSO is applied on 20 chromagents included in the first subgroup. PSO utilizes the concept of velocity. We assume the chromagents will move in a 159 dimensional space. The

Algorithm 6 *Genetic Algorithm*

procedure GeneticAlgo($GAPop$, $GAPopFitness$[:], $EliteCount$, $XoverFrac$)
$ElitePop$[:] $\Leftarrow$ best $EliteCount$ individuals in $GAPop$
$XoverPop$[:] $\Leftarrow$ Stochastic Uniform Selection from $GAPop$
$MutationaPop$[:] $\Leftarrow$all remaining chromagents
for $i = 1$ to $XoverFrac *$ size($GAPop$) **do**
  $XoverKids[i] \Leftarrow$ CrossOver($XoverPop[2 * i - 1]$, $XoverPop[2 * i]$)
end for
for $i = 1$ to size($MutationPop$) **do**
  $MutationKids[i] \Leftarrow$ Mutation($MutationPop[i]$)
end for
$GAPop[:] \Leftarrow [ElitePop[:], XoverKids[:], MutationKids[:]]$
return GAPop[:]
end procedure

procedure CrossOver($s1, s2$){$s1$ and $s2$ are parents}
for $i = 1$ to 159 **do**
  $x \Leftarrow$ random number in the range [0,1]
  if $x > 0.5$ **then**
    $s3[i] \Leftarrow s1[i]$
  else
    $s3[i] \Leftarrow s2[i]$
  end if
end for
return $s3$ {$s3$ is child chromagent produced from $s1$ and $s2$}
end procedure

procedure Mutation(s){s is a chromagent for mutation}
for $i = 1$ to 38 **do**
  $x \Leftarrow$ random number in the range [0,1]
  if $x > 0.5$ **then**
    $s[i] \Leftarrow s[i] + round(10 * x)$
  else
    $s[i] \Leftarrow s[i] - round(10 * x)$
  end if
end for
$x \Leftarrow$ random number in the range [0,1]
if $x > 0.5$ **then**
  $s[39] \Leftarrow s[39] + round(5 * x)$
else
  $s[39] \Leftarrow s[39] - round(5 * x)$
end if
for $i = 1$ to 20 **do**
  $x \Leftarrow$ random number in the range [0,1]
  if $x > 0.5$ **then**
    $s[40 + (i - 1) * 6] \Leftarrow$AND
  else
    $s[40 + (j - 1) * 6] \Leftarrow$OR
  end if
  for $j = 1$ to 5 **do**
    $y \Leftarrow$ random number in the range [0,1]
    $s[40 + j + (i - 1) * 6] \Leftarrow s[40 + j + (i - 1) * 6] + round(7 * y)$
  end for
end for
end procedure

159 genes of each individual act as the coordinate values for the respective 159 dimensions. Each individual remembers its best fitness value and the corresponding coordinates. The best fitness value of individual i is denoted by $pbest_i$. Each individual also knows the overall best fitness value chromagent that is denoted by *gbest* and its corresponding coordinates. We define a matrix M of dimensions 159 x 20. Each column of M corresponds to the current coordinate values of corresponding individual. We also maintain another matrix P of dimensions 159 x 20, which stores the coordinates of the corresponding best encountered fitness value individuals. A column vector G of length 159 stores the coordinate values corresponding to the best fitness value of each individual. Furthermore, each individual in the swarm of these 20 chromagents has its own velocity in each of the 159 dimensions. These velocities for every individual are stored in a matrix V of dimensions 159 x 20. Let us now consider an individual

Algorithm 7 *PSO Algorithm*

procedure PSOAlgo($PSOPop, PSOPopFitness[:], PersonalInc$, $GlobalInc$, training data)
for $i = 1$ to size($PSOPop$) **do**
  $M_x[:][i] \Leftarrow PSOPop[i]$
  $V_y[:][i] = V_x[:][i] + rand * PersonalInc * (P[:][i] - M_x[:][i]) +$ $rand$*$GlobalInc * (G[:] - M_x[:][i])$
  $M_y[:][i] = M_x[:][i] + V_y[:][1] * 1$ time unit
  $GlobalBestFittness \Leftarrow$ ComputeFitness($G[:]$, training data)
  $PersonalBestFittness \Leftarrow$ ComputeFitness($P[:][i]$, training data)
  $CurrentFittness \Leftarrow$ ComputeFitness($M_y[:][i]$, training data)
  if $CurrentFittness > PersonalBestFittness$ **then**
    $P[:][i] \Leftarrow M_y[:][i]$
  end if
  if $CurrentFittness > GlobalBestFittness$ **then**
    $G[:] \Leftarrow M_y[:][i]$
  end if
  $PSOPop[i] \Leftarrow M_y[:][i]$
  $x \Leftarrow y$
end for
return $PSOPop[:]$
end procedure

i (where $1 \leq i \leq 20$) and see how it changes its velocity in order to move towards the optimum point. The component of its velocity is maintained in i^{th} column of Matrix V. Its current location is stored in column i of matrix M. The objective of the chromagent is to adjust its velocity in such a fashion that it moves towards the optimal solution. This change in velocity is accomplished by the following equation:

$$V[:][i] = V[:][i] + rand * PersonalInc * (P[:][i] - M[:][i]) + rand * GlobalInc * (G[:] - M[:][i])$$

where $V[:][i]$ is a column vector consisting of all rows of matrix V but only the i^{th} column. Similarly $M[:][i]$ is a column vector from matrix M corresponding to the i^{th} chromagent, $PersonalInc$ and $GlobalInc$ are tuning parameters and their values are empirically determined to be 0.35 and 0.40 respectively.

In this way each of these chromagents gets a new velocity in order to move towards a better solution. This will complete one iteration and is equivalent to the production of a new generation in the GA. It is important to mention here that the larger value of $PersonalInc$ causes the chromagents to roam around the best fitness value that they had observed and larger $GlobalInc$ causes the chromagents to prematurely converge towards the best local maxima which an individual has seen. The smaller values of these parameters cause excessive wandering of chromagents and hence the convergence becomes very slow. However, the convergence is achieved with a reasonable exploration of the fitness landscape. The pseudo code of all this process is given in Algorithm 7.

2.6 Communication between GA and PSO

In order to efficiently evolve chromagents, it is important that the progress of GA should be communicated to PSO and vice versa. After every five iterations, we replace 1 worst individual of the GA group chromagents with the best individual of PSO group chromagents . Similarly, we also replace two worst individuals of PSO group with the two best of the GA group. This communication mechanism ensures that the hybrid system benefits from both PSO and GA. Note that an iteration in GA corresponds to generating the next population; while in PSO it corresponds to a velocity adjustment of all the chromagents. Algorithm 2 explains the communication in detail.

2.7 Stopping Criteria

The fitness values of chromagents improve with each iteration. In our experiments, we stop the simulations if any of the following criterion is met: (1) a total of 50 iterations have taken place, or (2) the fitness value of the best individual of PSO and GA groups has not changed in previous 25 iterations. Finally, the best chromagent among the population of 30 is chosen and is used as a rule base for the fuzzy system.

3. EXPERIMENTS: RESULTS AND COMPARISONS

Our definition of a real usable system should meet the following important requirements: (1) FAR and FRR below 5% (ideally it should be 0%, but practically 5% is enough), (2) user identification within a profile of 8 calls, and (3) the system should have a small processing overhead. We now use three types of analysis to satisfy the real usable requirements: (1) accuracy analysis, (2) scalability analysis, and (3) training and testing time analysis. In our experiments, we use the same training and testing methodology for all classifiers. During the training phase, we make one of the 10 users as the legitimate user and 2 of the remaining 9 as imposters. We train the system on their profiles. During the testing phase, however, we ensure that we never present an imposter to a classifier if it is trained even on one of his profiles. Consequently, it ensures that a system is able to identify an imposter without any a priori knowledge about his behavior. *We argue that in the real life it is not possible to get the profiles of potential imposters in advance.* Ideally, we should have a pure anomaly detection system, which does not take even the profiles of 2 imposters. We also show results for this scenario.

3.1 Accuracy Analysis

We have implemented and simulated our system in Matlab and compared its results with four classical classifiers – Naïve Bayes, Back Propagation Neural Networks, J48 Decision Tree, and fuzzy system – and three evolutionary schemes – fuzzy system optimized by Ant Colony Optimization (ACO), Particle Swarm Optimizer (PSO), and Genetic Algorithm (GA), individually. Note that Naïve Bayes, BPNN, and J48 were implemented in Weka [11]. We repeated our experiments for $ACO - Fuzzy, PSO - Fuzzy, GA - Fuzzy$, and $UGuard$ 500 times and the confidence interval turned out to be 95%. The results of experiments are reported in Table 3.

We can see in Table 3 that a simple fuzzy system and other classical machine learning algorithms are unable to meet our requirement of 5% average error rates. It is important to highlight that a random detection system provides 50% (on the average) FAR and FRR. The simple fuzzy system provides on the average 15.1% and 21.7% FAR and FRR respectively. The classical algorithms, Naïve Bayes, BPNN and J48, also provide on the average 11%, 12% and 24% FAR respectively while their FRR is approximately 6-7%. To conclude, none of them come close to our requirement of 5% FAR and FRR.

We now optimize the rule base of our fuzzy system with the help of well-known optimizers for dynamic environments:

Table 3: A comparative study of techniques on the basis of three features

	Naïve Bayes		BPNN		J48		Fuzzy System		ACO-Fuzzy		PSO-Fuzzy		GA-Fuzzy		UGuard	
Users	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
x1	10.1	5.32	14.2	6.34	21.3	6.97	13.2	31.3	8.13	8.01	7.44	8.21	7.22	6.31	**2.43**	**1.86**
x2	13.4	6.34	11.7	7.68	25.4	7.21	17.1	30.9	6.21	7.32	5.22	9.42	4.31	8.22	**1.71**	**1.92**
x3	11.2	4.73	12.3	7.21	27.9	5.34	20.6	21.8	11.7	8.51	8.23	7.22	6.34	9.31	**3.44**	**1.82**
x4	9.61	4.85	16.2	8.33	24.3	6.72	18.9	26.3	7.23	6.24	6.45	5.84	8.91	7.34	**1.19**	**1.38**
x5	10.4	5.92	9.56	9.22	28.1	7.43	11.4	17.8	4.34	6.33	3.97	4.92	**3.25**	5.81	3.37	**2.45**
x6	8.72	6.11	9.43	6.81	19.6	8.33	13.4	17.1	7.21	6.32	6.95	6.01	6.33	5.42	**2.31**	**3.11**
x7	13.1	7.23	17.1	7.13	22.3	9.12	11.2	18.7	9.92	4.52	9.12	**3.21**	8.93	3.71	**1.82**	3.34
x8	11.5	6.41	12.4	6.21	26.4	8.73	12.3	15.2	8.61	7.84	9.22	6.42	10.3	9.73	**1.01**	**1.72**
x9	12.7	5.34	14.1	8.33	22.1	7.12	16.1	13.4	5.21	6.34	5.02	6.17	5.61	6.29	**3.21**	**1.04**
x10	13.6	7.21	11.9	8.47	21.9	8.32	17.1	24.2	7.35	8.21	7.40	5.95	7.21	6.32	**2.40**	**1.33**
average	11.4	5.95	12.9	7.57	23.9	7.53	15.1	21.7	7.59	6.96	6.90	6.34	6.84	6.84	2.29	1.99
difference	9.11	3.96	10.6	5.58	21.6	5.54	12.8	19.7	5.30	4.97	4.61	4.35	4.55	4.85	-	-
improvement%	79.9	66.6	82.1	73.7	90.38	73.6	84.8	90.8	69.8	71.4	66.8	68.6	66.5	70.9	-	-

Ant Colony Optimizers (ACO), Particle Swarm Optimizers (PSO) and Genetic Algorithms (GA). An important consideration – other than the error rates – is that the selected optimizer must be simple and efficient so that it can be deployed on smart phones. Remember that smart phones have not only limited computing power and memory but also short battery life. One can see in Table 3 that using ACO, PSO and GA as an optimizer for the fuzzy system reduces the error rates on the average to 6-7% which is a significant improvement over existing systems. But if we use a hybrid of PSO and GA, the average error rates drop to approximately 2%. This justifies the use of hybrid PSO and GA as far as the error rates are concerned. The improvement in error rates is due to the combined use of feedback (inherent in PSO) and randomness (inherent in GA).

As mentioned earlier, we now show results of the systems, if they are used as anomaly detectors instead of classifiers. (An anomaly detector is trained on the profiles of the legitimate user only.) This scenario is more desirable in real-world. We tabulate the results in Table 4. One can see that this scenario presents a significant challenge to all classifiers. The error rates of non-hybrid optimizers drop to more than 25% which renders these systems totally useless on real smart phones. We report the results of top performing systems for this analysis. Other classifiers also provide an error rate of around 30% which have not been tabulated for brevity. However, UGuard – even in this challenging scenario – still provides on the average 10% error rates. This number is high albeit significantly better compared with other classification systems. In our future work, we want to focus on this situation to make our system usable even in this scenario.

It is to be noted that one profile of a user is made once he/she has made 8 calls and our analysis is done on the basis of profiles. We believe that two important questions might come to the mind of a careful reader: (1) what is the effect of varying the number of calls per profile on the accuracy of our system?, and (2) what is the relation between the number of profiles and the accuracy of our system?. We answer these questions under the heading of scalability analysis.

3.2 Scalability Analysis

Relationship between the size of a profile and the accuracy of our system:
We select three users x2, x4, and x8 for which our system provides the best FAR and FRR. We tabulate the results of our experiments in Table 5. It is obvious that the error rates significantly degrade once we reduce the number of calls per profile. Our results show that FAR and FRR does not significantly improve as we increase the number of calls per profile beyond 8 calls; therefore, we select 8 calls/profile.

Table 4: A comparative study of techniques as anomaly detection

	ACO-Fuzzy		PSO-Fuzzy		GA-Fuzzy		UGuard	
Users	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
x1	29.5	21.7	23.2	18.3	22.3	24.1	11.5	12.4
x2	25.2	21.2	25.1	21.2	26.1	27.3	12.4	11.2
x3	26.3	24.1	21.5	22.4	29.5	21.5	9.7	11.6
x4	27.3	25.1	21.4	21.3	24.4	28.4	10.1	10.1
x5	26.1	26.8	22.7	19.9	23.3	22.7	11.2	9.2
x6	28.9	19.7	21.9	21.1	26.5	21.9	9.4	9.1
x7	23.6	21.3	19.1	20.1	24.6	24.6	10.7	10.9
x8	27.1	21.3	20.6	20.7	26.4	31.7	9.8	11.3
x9	27.1	24.1	24.3	19.2	24.1	26.4	12.9	10.2
x10	26.1	25.1	20.6	21.7	31.7	23.1	11.5	10.4
Avg	26.7	23.0	22.0	20.6	25.9	25.2	10.9	10.6

Table 5: The relationship between accuracy of our system and the size of a profile

	number of calls constituting a profile							
	4		6		8		12	
Users	FAR	FRR	FAR	FAR	FAR	FAR	FAR	FAR
x2	11.2	12.1	6.12	7.24	1.71	1.92	1.52	1.73
x4	15.6	10.7	8.21	7.16	1.19	1.38	1.11	1.32
x8	13.8	11.3	7.33	9.32	1.01	1.78	0.99	1.75

The important reason for deteriorating FAR and FRR with less calls per profile is that the system does not capture enough information about the behavior of a user to accurately classify him.

Relationship between the number of profiles and the accuracy of our system:
We now investigate the minimum number of profiles of a user needed to achieve acceptable FAR and FRR. We again select x2, x4, and x8 users because we have relatively large number of profiles for these users. We tabulate the results of our experiments in Table 6. We conclude that the error rate of our system exponentially improves with an increase in the number of profiles of a user. From Table 6, we can say that a user is adequately protected once he has generated 9 profiles (4% error rates). However, if he generates 12 profiles then the error rates are dropped to less than 2%. The reason for

this behavior is that with the increasing number of profiles the rule base of our fuzzy system is optimized enough to understand the behavior of a user.

Table 6: The relationship between the accuracy of our system and number of profiles

Users	Number of profiles with 8 calls per profile							
	3		6		9		12	
	FAR	FRR	FAR	FAR	FAR	FAR	FAR	FAR
x2	15.7	13.2	8.32	10.3	4.13	4.63	1.71	1.92
x4	16.4	15.1	9.21	9.31	3.13	3.89	1.19	1.38
x8	14.1	17.2	8.01	10.1	2.89	4.23	1.01	1.78

3.3 Training and Testing Time Analysis

We now show the training (Train) and testing (Test) times of the classifiers in Table 7. Before getting into the details, it is important to mention here that the results reported for training and testing times are obtained from the simulations that have been done on a computer with 233MHz Intel processor and 32MB RAM. The mobile processors of smart phones have similar processing and memory specifications. (e.g. Nokia's N95 has a 330MHz processor and a 64MB RAM). The results are tabulated in Table 7. Fuzzy system does not have a training phase; therefore, its time is 0. All fuzzy-based systems have same testing time because these systems ultimately use its rule base for classification. Note that our system has 28 seconds of training time while the testing time is just 520 milliseconds. We retrain all classifiers, including our system, after every 5 profiles. The time to build 5 profiles is mostly in hours; therefore, we believe that it is acceptable to spend 28 seconds after few hours. During this time the response of a smart phone will slightly degrade. Once trained, we spend just 520 milliseconds for testing after every 8 calls.

Table 7: The processing overheads of classifiers on an old 233MHz, 32MB RAM computer

Algorithm	Train (*secs*)	Test (*secs*)	Algorithm	Train (*secs*)	Test (*secs*)
UGuard	28	0.52	**ACO-Fuzzy**	31	0.52
PSO-Fuzzy	21	0.52	**GA-Fuzzy**	35	0.52
Fuzzy system	0	0.52	**J48**	0.23	0.22
BPNN	4.8	2.0	**Naive Bayes**	0	0.13

4. RELATED WORK

It is important to note that most of the research in the domain of user authentication systems has been focussed on desktops and mobile phones have received very little attention. Moreover, the use of evolutionary algorithms are almost non-present for this purpose. Some of the relevant work on user authentication system has been reported in [3], [9] and [8]. However only [10] has done some preliminary work using GA. To the best of our knowledge no previous study has used any of the genetic algorithms for user authentication on smart phones.

5. CONCLUSION

In this paper, we propose our UGuard system in which a fuzzy classifier is optimized using a hybrid (PSO-GA) System. It takes the diffused input features' set as an input based upon the usage behavior of smart phones. We have evaluated our proposed system on a 8 day real dataset of 10 mobile users coming from different backgrounds. Our system is real usable because it correctly detects a legitimate user or an imposter with less than 2% average error rates (FAR and FRR). Hence, it satisfies the criteria of real usable system defined in Section 3. As a result, we suggest that our system has the potential to become an integral part of an OS of a smart phone that will not only help in securing sensitive information/data on a smart phone but also reduce the number of mobile phone thefts. In near future, we will focus our attention on converting our system to a pure anomaly detection which is only trained on the profile of a legitimate user.

6. REFERENCES

[1] Red herring mobiles scream for help: Uk-based mobile security company adds security to mobile phones, October 2006.

[2] B. Carse and T.C. Fogarty. A Fuzzy Classifier System Using the Pittsburgh Approach. In *Parallel Problem Solving from Nature–PPSN III: International Conference on Evolutionary Computation, the Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9-14, 1994: Proceedings*. Springer, 1994.

[3] M.T. Carta, B. Podda, and C. Perra. User Authentication Based on JPEG2000 Images. *Lecture Notes in Computer Science*, 3893:207, 2006.

[4] NL Clarke and SM Furnell. Authentication of users on mobile telephones–A survey of attitudes and practices. *Computers & Security*, 24(7):519–527, 2005.

[5] Ernst and Young Global Information Security Survey 2005 : Report on the Widening Gap.

[6] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[7] EH Mamdani et al. Application of fuzzy algorithms for control of simple dynamic plant. *Proc. IEE*, 121(12):1585–1588, 1974.

[8] JM McCune, A. Perrig, and MK Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *2005 IEEE Symposium on Security and Privacy*, pages 110–124, 2005.

[9] J. Rokita, A. Krzyzak, and CY Suen. Cell Phones Personal Authentication Systems Using Multimodal Biometrics. *Lecture Notes in Computer Science*, 5112:1013–1022, 2008.

[10] K. Sung and S. Cho. GA SVM Wrapper Ensemble for Keystroke Dynamics Authentication. *Lecture Notes in Computer Science*, 3832:654, 2006.

[11] I.H. Witten, University of Waikato, and Dept. of Computer Science. *WEKA Practical Machine Learning Tools and Techniques with Java Implementations*. Dept. of Computer Science, University of Waikato, 1999.

[12] L.A. Zadeh. Fuzzy sets. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers*, 1996.