5

Enhanced Permission Based Malware Detection in Mobile Devices Using the Proposed MSGP-MS Method

5.1 Introduction

To ensure security in mobile devices, an enhanced iris biometric authentication method is effectively discussed in chapter 4. Smartphone usage has been rapidly increasing and its popularity makes the attackers more attracted to these devices. This has made smartphones more vulnerable to malware attacks and a target for information and identity theft. Malwares can be easily installed on mobile devices using a malicious application with the intention of breaching device security policy with respect to confidentiality, integrity and availability of data.

With the rapid growth of malware attacks in mobile devices, the detection of malware becomes more critical. A variety of researches have been developed to defense malware, but the mobile device users continuously suffer private information leak or economic losses due to malwares. The malicious applications can be distributed to mobile devices through an application market. Mobile malware detection schemes are categorized into three groups: static analysis, permission analysis, and dynamic analysis. To be specific, permission based detection techniques extract security configurations and check the applications against its installation. These third-party applications that have the capability of accessing resources such as phone hardware, settings, user data, and others through permissions.

The proposed framework intends to develop an optimized machine learning classifier based malware detection scheme on mobile devices to detect malicious applications and to enhance security and privacy of smartphones. In this chapter, the combined algorithm namely, MSGP-MS which is a combination of Random Forest with Particle Swarm Optimization is used to detect the mobile malware in mobile applications. The malware detection scheme identifies the presence of malware in android applications and it enhances security of smartphones. This scheme is based on a permission framework for analysing and classifying android applications using supervised learning techniques.

5.2 Steps of Proposed Contribution Two - MSGP-MS Method

The objective of the contribution two MSGP-MS Method is to enhance Malware detection in android applications using Optimized Machine Learning Classifiers. The proposed contribution consists of four steps and are discussed below.

- Data Collection
- Permission extraction and selection
- Classification
- Optimization for Malware detection

To detect the presence of malware in android applications and to enhance the security of mobile device and data, a combined method called MSGP-MS is proposed. MSGP-MS classifier is the combination of Random Forest with Particle Swarm Optimization algorithm. Figure 5.1 shows the four different steps of proposed contribution two MSGP-MS Classifier.

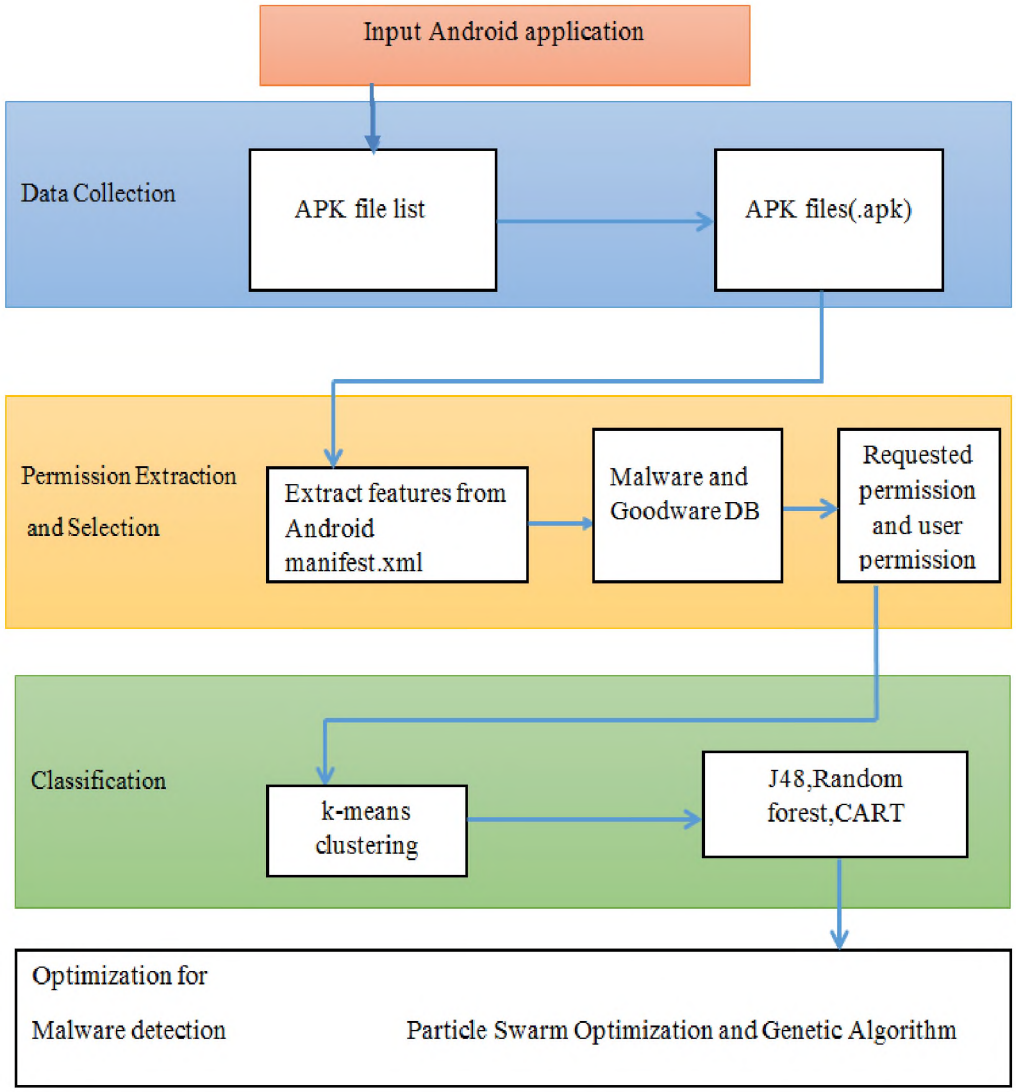

Figure 5.1 Block Diagram of Proposed Contribution MSGP-MS Method

5.2.1 Data Collection

The dataset consists of a total of 200 applications split into malicious applications and goodware applications. There are no duplicate applications in the dataset and every sample is labelled as benign or malicious. These datasets can be viewed in android application package (APK) list. The APK list contains numerous apk files for further permission extraction and feature selection.

5.2.2 Permission Extraction and Selection

Features are the attributes used for defining the permission characteristics of an application. For any downloaded Android application, retrieves the features from the corresponding application package file. Analyse the Manifest file of an application and identify real permissions required by the application. The values of selected features are stored as a feature vector, which is represented as a sequence of bits (0's or 1's). A feature set can be specified as a feature vector, which includes all the permissions that are requested from the user.

Every application must have an android Manifest.xml in its root directory. The manifest presents essential information about the application to the Android system. The features in each Android application are extracted through the following steps:

- Download the goodware and malware applications available. Decompress the application (.apk) file by the reengineering process and separate it into its component files.
- One among the files is the Android Manifest.xml file. This xml file has various permissions contained in it. The permissions of the XML file are extracted and converted into binary form (0 or 1).
- The binary bit of the feature is set valid (1) if the permission is present in the apk file else the bit is set as invalid (0). These permissions form the features through which the dataset is built. The few sample permissions are described in Table 5.1.

Table 5.1 List of Permissions on an APK file

Permission	Usage
Android. permission. PROCESS_OUTGOING_CALLS	The application allows the user to see the number being dialled during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
Android. permission. RECEIVE_SMS	Allows an application to monitor incoming SMS messages, to record or perform processing on them.
Android. permission. SET_PROCESS_LIMIT	Allows an application to set the maximum number of (not needed) application processes that can be running.
Android. permission. CALL_PHONE	Allows an application to initiate a phone call without going through the Dialler user interface for the user to confirm the call being placed.

Finally, the dataset is formed which is saved in a text format is shown in figures 5.2 and 5.3.

0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, goodware

Figure 5.2 Sample dataset of a Goodware Application

0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, malware

Figure 5.3 Sample dataset of a Malware Application

Feature selection method is used for reducing the dimension of a dataset by removing the features (attributes) which are not beneficial to be used in the analysis. Efficient feature selection method introduces performance gains by reducing the dataset size and the time spent in classification analysis. It is the method of determining the rank of appropriate features through the entropy difference between the cases of accurate classification of features. The feature selection is done based on the gain ratio. The features with a higher gain ratio, yield better optimality to the resultant generation. The features are selected based on the Gain value by referring whether they are greater than '0' and only the features which are greater than '0'are included in the minimized dataset or selected features. Gain of an attribute A for a sample S is given in equation (5.1)

$$Gain(S, A) = Entropy\ (S) - \sum_{V \in Values(A)|S|} |SV|\ Entropy\ (SV) \qquad (5.1)$$

5.2.3 Classification Techniques for Malware Detection

Decision tree classifiers are tree based classifiers for instances represented as feature vectors. They recursively partition the set of records and use a depth first greedy method or breadth first approach. Nodes specifies test features, the branch specifies each value of the feature and the leaves specify the category until all the data items belong to a particular class. Decision Trees based classification of instances by sorting feature vectors. Three machine learning classification algorithms are applied to the data sets: J48, Classification and Regression Tree (CART) and Random Forest (RF). Before applying classification algorithms, clustering is done using K-means algorithm with the dataset.

5.2.3.1 K-means Clustering

K-means clustering is a non-parametric supervised learning algorithm used for classification. K-means clustering is a data driven method. This method relatively assumes the distributions of the underlying data which guarantees the convergence of clusters on large datasets. Clustering is performed on training instances to obtain 'k' disjoint clusters. Each cluster represents a region of similar instances in terms of Euclidean distances between the instances and their cluster centroids. The Euclidean distance for real values is calculated using equation 5.2,

$$E = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{5.2}$$

The *k*-means clustering aims to partition the *n* observations into $k (\leq n)$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares. The average output value among neighbours is defined in equation 5.3,

$$arg_s min \sum_{i=1}^{k} \sum_{x \in Si} ||x - \mu_i||^2 \tag{5.3}$$

5.2.3.2 J48

J48 builds decision trees from a set of labelled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree telling to choose that class. However, it can also happen that none of the features give any information gain. Information gain IG(A) is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S is reduced after splitting set S on attribute A. Equation 5.4 defines information gain,

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t) \tag{5.4}$$

In this case, J48 creates a decision node higher up in the tree using the expected value of the class. J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. Further it provides an option for pruning trees after creation.

5.2.3.3 Classification and Regression Tree

Tree models where the target variable can take a finite set of values are called classification trees. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Classification and Regression Trees use cross validation or a large independent test sample of data to select the best tree from the sequence of trees considered in the pruning process. The basic CART building algorithm is a greedy algorithm that chooses the locally best discriminatory feature at each stage in the process. In the implementation of CART, the dataset is split into the two subgroups that are different ones with respect to the outcome. CART partitions the feature space into a set of rectangles and fit a simple model in each one. Then it constructs a binary tree structured classifiers by repeated splits of subsets of the measurement spaces into two descendant subsets. This method assigns a class label for each terminal subset and the resulting partition of x corresponds to the classifier.

The Gini measure is the measure of impurity of a node and is commonly used when the dependent variable is a categorical variable, defined in equation 5.5 as,

$$g(t) = \sum_{j \neq i} p\left(\frac{j}{t}\right) p\left(\frac{i}{t}\right) \tag{5.5}$$

The entropy is defined in equation 5.6,

$$(f) = -\sum_{i=1}^{rn} f_i \log_2 f_i \tag{5.6}$$

5.2.3.4 Random Forest

Random Forest, a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution of all trees in the forest. RF are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Each tree is independently constructed using a bootstrap sample of data.

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This OOB (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest.

It is also used to get estimates of variable importance. After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

A relationship between random forests and the nearest neighbour algorithm is defined in equation 5.7,

$$y = \sum_{i=1}^{n} \left(\frac{1}{m} \sum_{j=1}^{m} Wj(Xi, X)\right) Yi \qquad (5.7)$$

here, $W_j(Xi,X)$ is the non-negative weight of the i^{th} training point relative to the new point *x*.

5.2.4 Optimization Techniques for Malware Detection

Current techniques in malware classification do not give a good classification result when it deals with the new and unique types of malware. For this reason, the usage of optimization techniques, namely Genetic Algorithm and Particle Optimization Algorithms are used to optimize the malware classification system. This new malware classification system has an ability to train and learn by itself, so that it can predict the current and upcoming trend of malware attacks.

5.2.4.1 Genetic Algorithm

Evolutionary Algorithm includes the survival of the fittest idea into a search algorithm which provides a method of searching, which does not need to explore every possible solution in the feasible region to obtain a good result. The basic process for a genetic algorithm

- Initialization - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
- Evaluation - Each member of the population is then evaluated and a 'fitness' value is calculated for individual. The fitness value is calculated by best fits with desired requirements. These requirements could be simple, 'faster algorithms are better', or more complex, 'stronger materials are better but they shouldn't be too heavy'.

- Crossover - During crossover, new individuals are created by combining aspects of selected individuals. By combining certain traits from two or more individuals creates an even 'fitter' offspring which will inherit the best traits from each of its parents. The crossover is defined in equation 5.8.

$$\text{Crossover } n = (\alpha - n)/2 \tag{5.8}$$

- Mutation - Mutation typically works by making very small changes at random to an individual's genome.

- Selection - Constantly improving populations overall fitness.

5.2.4.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. In the PSO algorithm each individual is called a "particle", and is subject to a movement in a multidimensional space that represents the belief space. There is no restriction for particles to share the same point in belief space, but in any case their individuality is preserved. Each particle's movement is the composition of an initial random velocity and two randomly weighted influences: individuality, the tendency to return to the particle's best previous position, and sociality, the tendency to move towards the neighbourhood's best previous position. The initialize particle velocity is defined in equation 5.9,

$$Vi = V(-|b_{up}-b_{lo}|,|b_{up}-b_{lo}|) \tag{5.9}$$

where 'up' and 'lo' are the upper and lower boundaries of search space.

The best fitness particle velocity with according to velocity equation is shown in equation 5.10,

$$V_0^i = \frac{xmin+rad(xmax+xmin)}{\Delta t} \tag{5.10}$$

5.3 Flow diagram of the Proposed Contribution Two - MSGP-MS Method

The proposed MSGP-MS method enhances the detection of Malware in android applications using Optimized Machine Learning Classifiers such as Random forest with Particle Swarm Optimization. The flow diagram of the proposed contribution two is shown in figure.5.4.

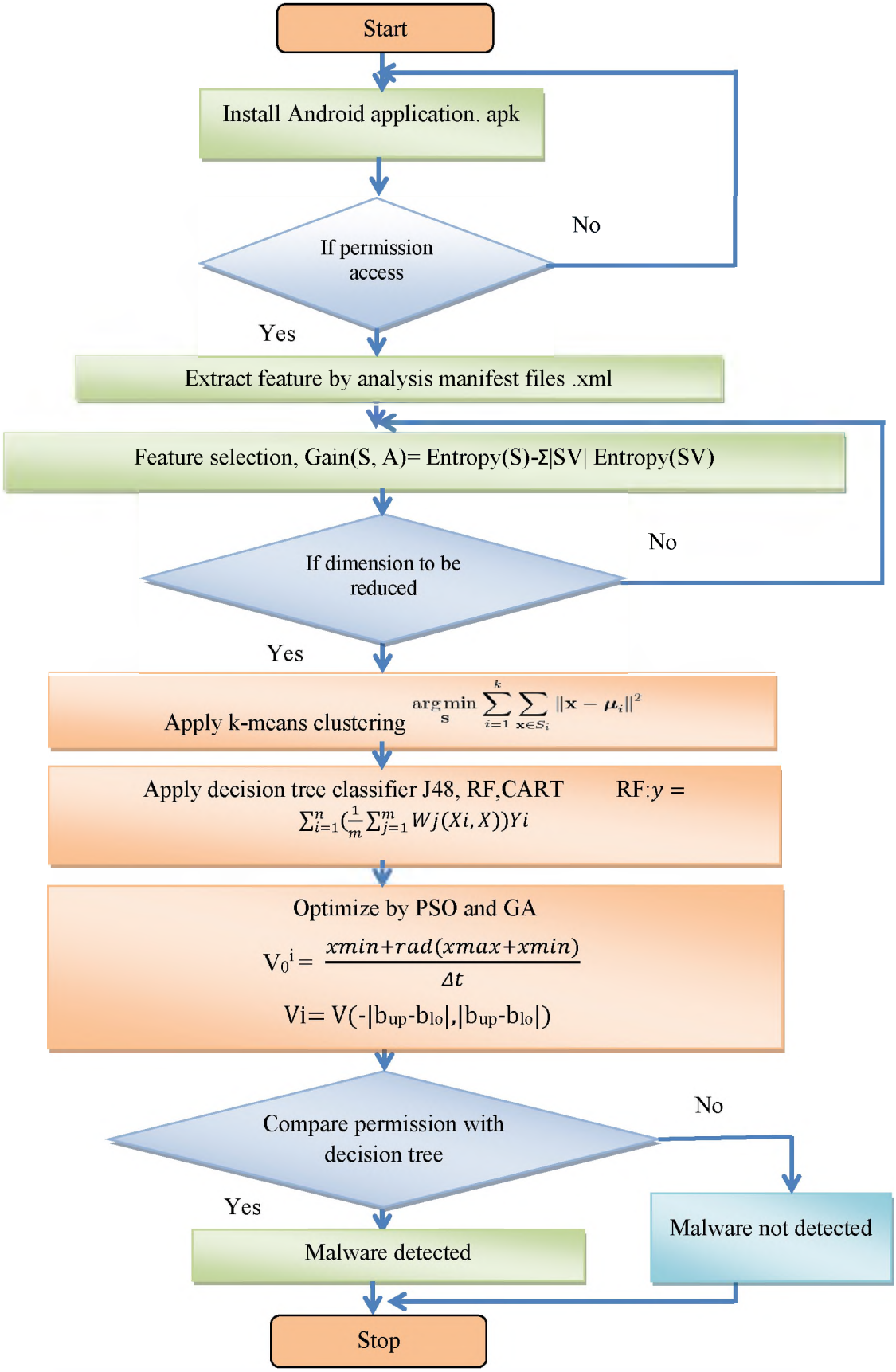

Figure 5.4 Flow chart of Proposed MSGP-MS Method

5.4 Steps involved in the Proposed MSGP-MS Method

The steps used for the enhanced Malware detection in android applications using Optimized Machine Learning Classifiers are discussed below:

Step 1: Data collection of android applications which consists of normal and malicious

Step 2: Permission are extracted, labelled and stored to the feature vector.

Step 3: Apply gain entropy for feature selection and characterizing the behaviour of each application.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{V \in \text{Values}(A)} \frac{|SV|}{|S|} \text{Entropy}(SV)$$

Step 4: Apply k-means cluster for dimension reduction and the average output value among neighbours is defined as

$$\arg_s \min \sum_{i=1}^{k} \sum_{x \in Si} ||x - \mu_i||^2$$

Step 5: Apply supervised learning classifier such as J48, Random Forest and Classification and regression tree (CART) to produce several detection model. The entropy can be defined as,

$$\varphi(p) = -\sum_k = p_k \log p_k$$

Step 6: Apply optimization algorithm such as genetic algorithm and particle swarm optimization with the best classifier to detect the presence of malware in the application.

Six steps discussed above are followed to detect the presence of malware in applications and provide enhanced security for data and mobile device. The algorithm proposed is discussed below.

5.5 Pseudo code of MSGP-MS Method

The algorithmic procedures applied for classification and optimization to enhance malware detection and security by the proposed approach MSGP-MS is shown in table 5.2 and table 5.3 below.

Table 5.2 Pseudo code of MSGP-MS Method using PSO with RF classifier

For each particle i = 1, ..., S do:
Randomly sample the training data Dwith replacement to produce i D
Create a root node, i N containing i D
Call BuildTree(i N)
end for
Build Tree
if N contains instances of only one class then
return
else
Randomly select x% of the possible splitting features in N
Select the feature F with the highest information gain to split on
Create f child nodes of N , 1 N ,., f N , where F has f possible values (1F ,. , fF)
for i = 1 to f do
Set the contents of i N to i D , where i D is all instances in N that match i F
Initialize the particle's position with a uniformly distributed random vector: $x_i \sim U(b_{lo}, b_{up})$, *where* b_{lo} *and* b_{up} *are the lower and upper boundaries of the search-space.*
Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
if $(f(p_i) < f(g))$ *update the swarm's best known position:* $g \leftarrow p_i$
Initialize the particle's velocity: $v_i \sim U(-|b_{up}-b_{lo}|, |b_{up}-b_{lo}|)$
Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:
end if
end for
for each particle i = 1, ..., S do:
for each dimension d = 1, ..., n do:
Pick random numbers: $r_p, r_g \sim U(0,1)$
Update the particle's velocity: $v_{i,d} \leftarrow \omega\, v_{i,d} + \varphi_p\, r_p\, (p_{i,d}-x_{i,d}) + \varphi_g\, r_g\, (g_d-x_{i,d})$
Update the particle's position: $x_i \leftarrow x_i + v_i$
end for
end for
if $(f(x_i) < f(p_i))$ *do:*
Update the particle's best known position: $p_i \leftarrow x_i$
if $(f(p_i) < f(g))$ *update the swarm's best known position:* $g \leftarrow p_i$
end if
end if

Table 5.3 Pseudo code of MSGP-MS Method using GA with RF classifier

```
size α of population,
rate γ of mutation,
number δ of iterations
generate α feasible solutions randomly;
save them in the population
        Randomly sample the training data Dwith replacement to produce i D
        Create a root node, i N containing i D
        Call BuildTree( i N )
end for
Build Tree
if N contains instances of only one class then
        return
else
        Randomly select x% of the possible splitting features in N
        Select the feature F with the highest information gain to split on
        Create f child nodes of N , 1 N ,., f N , where F has f possible values (1F ,. , fF )

Initialization
generate α feasible solutions randomly;
save them in the population PoP; Loop until the terminal condition
select the best N solutions in POP and save them in PoP1;
Crossover
number of crossover n=(α− n)/2 ;
for j=1 to n do
randomly select two solutions XA and XB from POP ;
generate XC and XD by one-point crossover to XA and XB;
save XC and XD to POP2 ;
endfor
 Mutation
for j=1 to n do
select a solution Xj from POP2;
mutate each bit of Xj under the rate γ and generate a new solution Xj' ;
if Xj' is unfeasible
update Xj' with a feasible solution by repairing Xj' ; endif
update Xj with Xj' in POP2;
endfor
Updating
update POP= POP1+ POP2 ;
endfor
```

5.6 Experimental Setup and Results

In the experimentation, the collected android applications are stored in apk file list. These applications are either normal or malicious one. The features are extracted based on permission request from the android manifest files (.xml). The rooting of malware is identified based on the classifier and optimization algorithms used. A sample of some Goodware and Malicious applications used as labelled dataset for experimentation is shown in Annexure II. The proposed MSGP-MS method is implemented. The experimentation methodology is shown in figure 5.5.

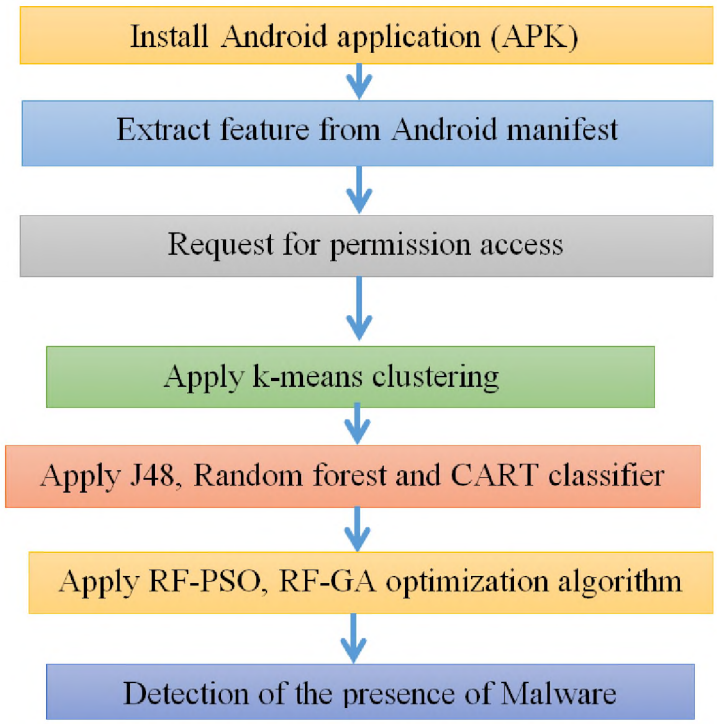

Figure 5.5 Experimentation Methodology for Contribution MSGP-MS Method

To evaluate the performance of the proposed method, the following five parameters listed below are used and defined:

i. True Positive Rate (TPR)
ii. False Positive Rate (FPR)
iii. Accuracy
iv. Precision Value
v. Recall Value

i. True Positive Rate (TPR)

The true positive rate is defined as percentage of correctly identified good ware applications. It is calculated by the below equation 5.11

$$TPR = \frac{True\ Positive}{True\ Positive+False\ Negative} \quad (5.11)$$

ii. False Positive Rate(FPR)

The ratio of incorrectly identified malicious applications. It is defined by equation 5.12,

$$FPR = \frac{False\ Positive}{True\ Positive+False\ Negative} \quad (5.12)$$

iii. Accuracy

Accuracy calculated by the equation 5.13 showned below

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \quad (5.13)$$

where True positive is correctly identified, False positive is incorrectly identified, True negative is correctly rejected and False negative is incorrectly rejected where TP is the Number of True Positive Instances, FN is the Number of False Negative Instances, FP is the Number of false Positive Instances, and TN is the Number of True Negative Instances. Authentication accuracy indicates the possibility of correctly identifying an individual (including both imposters and legitimate users).

iv. Precision Value

It is the ratio of retrieved instances that are relevant. It is also called positive

Predictive value. Precision reflects the fraction of correctly classified instances within all classified instances. Precision is a different measure than the accuracy of classification. It is defined in equation 5.14,

$$Precision = \frac{True\ Positive}{True\ Positive+False\ Positive} \quad (5.14)$$

v. Recall

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. It is defined by equation 5.15,

$$Recall = \frac{True\ Negative}{True\ Negative+False\ Negative} \quad (5.15)$$

Table 5.4 provides the comparison of parameters between J48, CART and Random Forest. The given parameters are True positive rate, false positive rate, Precision Value in (%) and Recall Value in (%) and Accuracy in (%).

Table 5.4: Performance Comparison Results of Classifiers

Methods	TP rate (%)	FP rate (%)	Precision (%)	Recall (%)	Correctly Identified Instances (%)	Incorrectly Identified Instances (%)
J48	0.83	0.17	0.87	0.79	83.3	16.7
CART	0.79	0.21	0.86	0.69	79	21
Random Forest	**0.87**	**0.13**	**0.91**	**0.81**	**86.8**	**13.2**

Figure 5.6 gives the comparison of experimental results. It is observed that random forest has high correctly identified instances of about 86.8% when compared to J48 whose correctly identified instance is 83.3% and CART of correctly identified instance 79%. Again to increase this accuracy, optimization techniques are used.

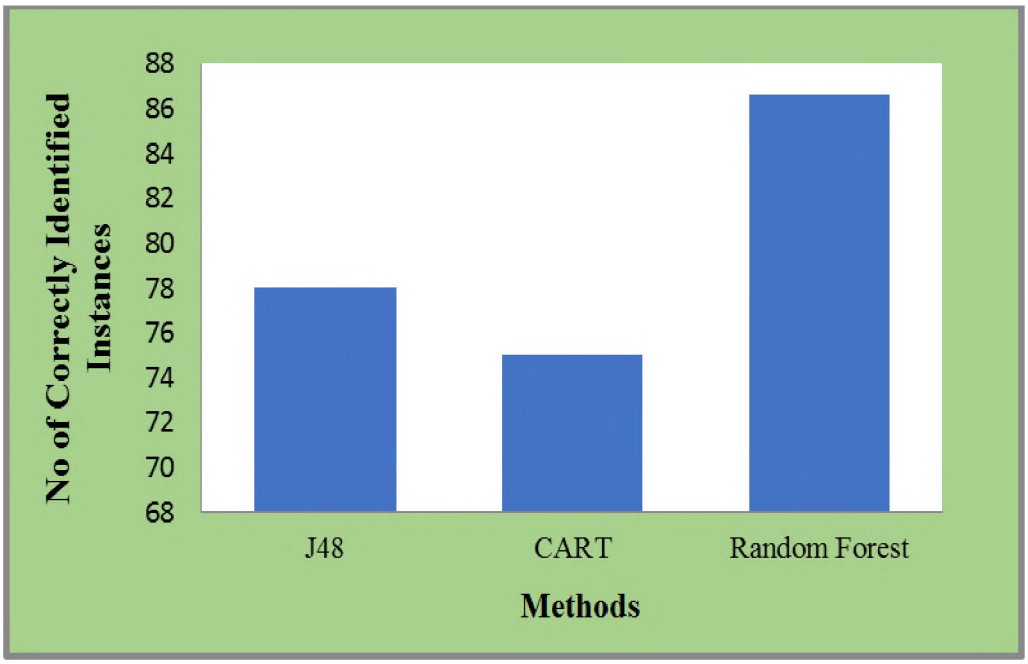

Figure 5.6 Comparision results of Classifiers for Contribution Two

Table 5.5 provides the comparison of Random Forest, Genetic Algorithm and Particle Swarm Optimization using the parameters such as correctly identified instances (Accuracy) in % and incorrectly identified instances in %.

Table 5.5: Performance Comparison of Optimized classifier results for Proposed MSGP-MS Method

Parameters / Methods	Correctly Identified Instances (%)	Incorrectly Identified Instances (%)
Random Forest	86.8%	13.2%
Genetic Algorithm with RF	87%	13%
Particle Swarm optimization with RF	**88.4%**	**12.6%**

Figure 5.7 gives the comparison, that particle swarm optimization has high correctly identified instances of about 88.4% when compared to genetic algorithm of correctly identified instance is 87% and random forest of correctly identified instance 86.8%.

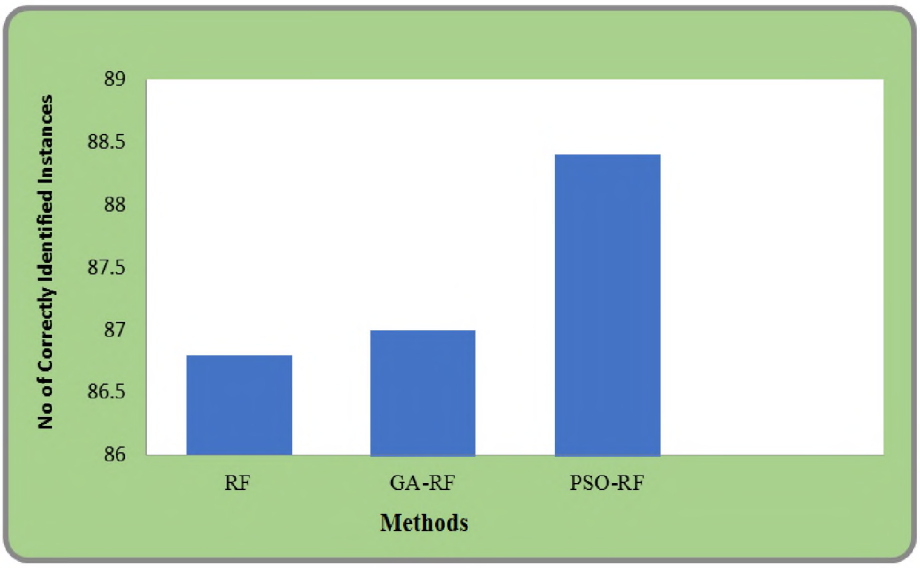

Figure 5.7 Comparison of Optimized classifier results for Proposed MSGP-MS Method

5.7 Chapter Summary

In this chapter, the proposed framework detects the presence of malware in the mobile device applications using optimized machine learning classifiers is discussed. MSGP-MS classifier, a combination of Random Forest algorithm with particle swarm optimization has high correctly identified instances of about 88.4% when compared to RF with genetic algorithm of correctly identified instance is 87% and random forest of correctly identified instance 86.8%. The malicious applications in mobile devices are detected effectively and ensures mobile device security. On the other hand, a mobile device is still a resource constrained one, due to the limitations such as low storage, less security and few applications generally demand more resources than a mobile device can pay for. To secure outsourcing of mobile device data over cloud is proposed using Hybrid Cryptographic algorithms in contribution three and explained in chapter 6.