Improving Reputation Systems for Wireless Sensor Networks using Genetic Algorithms

Zorana Banković, David Fraga, Juan Carlos Vallejo, José Manuel Moya
Department of Electronic Engineering, Technical University of Madrid
Av. Complutense 30, 28040 Madrid, Spain,(+34) 915495700, ext. 4223, 4227, 4224
{zorana, dfraga, jcvallejo, josem}@die.upm.es

ABSTRACT

In this article we propose to couple reputation systems for wireless sensor networks with a genetic algorithm in order to improve their time of response to adversarial activities. The reputation of each node is assigned by an unsupervised genetic algorithm trained for detecting outliers in the data. The response of the system consists in assigning low reputation values to the compromised nodes cutting them off from the rest of the network. The genetic algorithm uses the feature extraction process that does not capture the properties of the attacks, but rather relies on the existing temporal and spatial redundancy in sensor networks and tries to detect temporal and spatial inconsistencies in the sequences of sensed values and the routing paths used to forward these values to the base station. This solution offers many benefits: scalable solution, fast response to thwart activities, ability to detect unknown attacks, high adaptability, and high ability in detecting and confining attacks. Comparing to the standard clustering algorithms, the benefit of this one is that it is not necessary to assign the number of clusters from the beginning. The solution is also robust to both parameter changes and the presence of large amounts of malicious data in the training and testing datasets.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General – *Security and Protection;* C.2.1 [**Computer Communication Networks**]: Network Communication and Design – *Wireless Communication;* I.2.6 [**Artificial Intelligence**]: Learning; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Performance, Design, Reliability, Experimentation, Security.

Keywords

Wireless sensor networks; reputation system; unsupervised genetic algorithm.

1. INTRODUCTION

Technological advances achieved in the previous decade have paved the way for the development and deployment of wireless sensor networks (WSN). Their development was mainly motivated by military applications, but over the years their deployment has been introduced to other areas, i.e. control and monitoring in industry, environment, health, etc.

WSNs consist of a huge number of sensor nodes. For this reason, the nodes have to be very cheap, so they exhibit very limited power and computational resources, small memory size and low bandwidth usage. The nodes within a WSN are densely deployed in the area or the phenomenon to be observed, thus providing high level of redundancy, which can serve as a way to discriminate the erroneous nodes. Furthermore, WSNs are often deployed in unattended or even hostile environments, making their securing even more challenging. However, due to the limited resource, the implementation of complicated security techniques is impossible. For all these reasons, the security of these networks is very weak.

The usage of reputation systems has been proposed as a feedback mechanism in order to take advantage of the existing spatial and temporal redundancy and thus discard faulty or manipulated data [1]. Since the collective opinion in a community determines an object's reputation score, reputation systems represent a form of collaborative sanctioning and praising. A low score represents the collaborative sanctioning of an object that the community perceives as having or providing low quality, and vice versa. Reputation scores change dynamically as a function of incoming ratings. A high score can quickly be lost if rating entities start providing negative ratings, just as it is possible for an object with a low score to recover and regain a high score.

Extensive research has been done on modeling and managing trust and reputation. It has been demonstrated that rating trust and reputation of individual nodes is an effective approach in distributed environments to improve security, support decision-making and promote node collaboration. To enhance the security of reputation systems, a set of unsupervised learning algorithms [2] has been proposed to detect statistical anomalies in the environment, and to feed refined trust information back to the reputation systems [3]. The main idea is to detect attacks in their early stages, and in this way enhance the response time of reputation systems. In this work we present in more detail unsupervised genetic algorithm (GA) used to this end. Previous solutions, such as SOM [3], have an important drawback, which is the need to set the number of clusters from the beginning, when we do not know its optimal value. SOMs are also highly sensitive to both parameter changing and the level of presence of malicious data during the training. In order to overcome these issues, we propose a GA that in essence searches for an optimal clustering, thus the number of clusters does not have to be set from the beginning. GAs are also known for their robustness.

The rest of the work is organized as follows. Section 2 provides an overview on the existing solutions. Section 3 details the proposed solution, while Section 4 presents developed GA. The approach is evaluated in Section 5, and conclusions are drawn in Section 6.

2. DECISION FRAMEWORK BASED ON TRUST AND REPUTATION

Trust and reputation have recently been suggested as an effective security mechanism for open and distributed environments (Ad Hoc networks [4], WSNs [3, 4], P2P networks [5], etc.). In essence, the nodes that do not behave properly (according to the established policy of "proper" behavior) will have low reputation, so the rest of the nodes will avoid any collaboration with them, which is equivalent to its isolation from the network. There are many different definitions of trust and reputation, but in essence trust is a belief about future behavior that one node holds in others and it is based on its own experience, thus its main characteristic is subjectivity. On the other hand, reputation is considered to be the global perception of the behavior of a node based on the trust that others hold in it, thus considered to be objective [4]. Alternatives to reputation systems can be incentive systems [6], where it is advantageous for the nodes to act in a way that the resulting global welfare is optimal.

Regarding the detection and protection from attacks, our idea is to avoid cryptography [7], as due to open environment in WSNs side-channel attacks [8] that can be used to guess the secret keys become an important threat. We also want to avoid detection using information such as location [9] that can easily be forged.

On the other hand, machine learning is very convenient in the situations where we have to deal with noisy and incomplete information, and when the requirements are often fuzzy and incomplete, such as in security. Furthermore, its advantages when used in network security [10, 11] are already proven. However, instead of learning typical behavior of attacks [13], which can help us only in detecting known attacks, or learning signatures of normal data [12] whose number if huge, we base our approach on the detection of outliers and characterize sensor outputs and routing paths using their temporal and spatial sequences. Furthermore, we do not have to label clusters [10] as normal or anomalous. This permits us to avoid pre-processing the training data, and also provides the possibility to detect unknown attacks.

We further propose to couple a reputation system with a genetic algorithm for detecting intrusions, where the reputation values are based upon the decision made by GA without using second-hand information. Reputation takes values from 0 to 1, 0 standing for no trust in the entity and 1 for the absolute trust. In essence, GA examines temporal and spatial coherence of the nodes, i.e. detects sharp changes that should not occur during the normal processing, and decreases the reputation values of nodes that exhibit suspicious behavior. In this way, the reaction of the intrusion detection system consists in assigning low reputation to suspicious nodes which cuts them off from the rest of the network, as rest of the nodes will avoid any kind of interaction with low-reputation nodes. Regarding redemption and secondary response, we adopt the following idea: as soon as a node exhibits suspicious behavior, its reputation is rapidly decreased; on the other hand, if a malicious node starts behaving properly, its reputation increases for small amounts until it has been behaving properly for sufficient period of time. This serves as a sort of protection from selective behavior, since the nodes have to behave properly most of the time in order to be considered as good.

Our approach offers many advantages. Since it does not use reputation information from other nodes, it is resilient to bad-mouthing attack which is one of the main vulnerabilities of standard reputation systems. Furthermore, it confines attacked nodes faster while relying on much lower node redundancy than standard reputation systems [3].

3. PROPOSED SOLUTION

3.1 Envisioned WSN Model

We envision WSNs (Fig.1) where most of the sensors exhibit limited resources, but there are also a number of PDA-like sensors with more resources (comparable to the resources of a PDA). Their number is significantly smaller than the number of the "normal" sensors, at least an order of magnitude smaller. There is at least one base station. The nodes can organize themselves either in a hierarchical or flat manner. Nodes can be fixed or mobile, although it is assumed that the majority of the nodes are fixed.

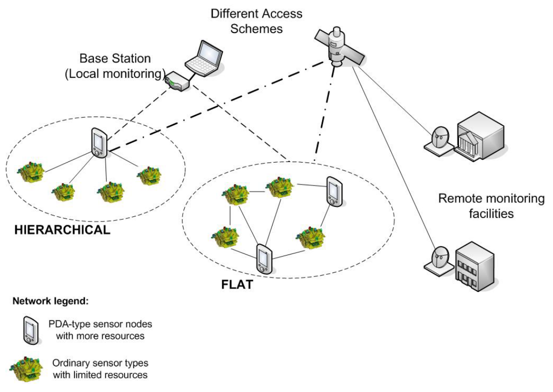
Different Access Schemes
Base Station (Local monitoring)
Remote monitoring facilities
HIERARCHICAL
FLAT
Network legend:
PDA-type sensor nodes with more resources
Ordinary sensor types with limited resources

Figure 1. WSN Model

3.2 Attack Assumptions

In order to provide uninterrupted network operation, core network protocols (aggregation, routing and time synchronization) have to be secured. Regarding the attacks on the aggregation protocol [14], we assume that they demonstrate themselves in skewed aggregated values, which can be the result of either a number of skewed sensed values, or a compromised aggregated node. The assumption is very reasonable, having in mind that the main objective of these attacks is to provide wrong picture of the observed phenomenon.

On the other hand, in time critical systems it is mandatory to receive information within certain time window. If the attacker manages to introduce delays or desynchronize clock signal in various nodes, the received critical information will not be up to date, which can destabilize the system. Also, if the received information is not up to date, the aggregated value will be skewed, as it will also be out of date. For these reasons, and given the existing redundancy in WSNs, we believe that these attacks can be detected as temporal and/or spatial inconsistencies of sensed values.

Regarding attacks on routing protocols [14], we assume that they will introduce new and different paths than those that have been seen before. Here we have attacks whose main objective is to compromise the routing protocol, and they usually do it by spoofing or altering the data stored in the routing tables of the nodes. In this way, the resulting routing paths will be different from those used in a normal situation. In the case of wormhole for example, two nodes that are not within each other's radio range result in consecutive routing hops in routing paths, which is not possible in a normal situation. Thus, the assumption about the attacks resulting in routing paths different from those that appear in normal situation is reasonable. In this case we want to detect temporal inconsistencies in paths used by each node.

3.3 Feature Extraction and Formation of Model

The main idea of our work is to find temporal and/or spatial inconsistencies in the sequences of sensed values and routing paths, as it is very probable that it will be the result of an attack on the core network protocols. Thus, we want to provide the model of the data that would capture these properties and map it to a vector space, which would allow us to deploy machine learning.

For the case of sensed values, we follow the idea of *n*-grams [15]. We will illustrate this with a short example for a sensor that detects presence. Let the sensor give the following output during the time window of size 20: 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0. If we fix the n-gram size on 3, we extract all the sequences of size 3 each time moving one position forward. In this way we can observe the following sequences and the number of their occurrences within the time window: 111 – occurs 6 times, 110 – 2, 100 – 2, 000 – 6, 001 – 1, 011 – 1. Thus, we can assign them the following sequences: 111 – 0.33, 110 – 0.11, 100 – 0.11, 000 – 0.33, 001 – 0.06, 011 – 0.06. In our model, the sequences are the features and their frequencies are the corresponding feature values. Thus, the sum of the feature values is always equal to 1. In our algorithm this characterization is performed in predefined moments of time and takes the established amount of previous data, e.g. we can perform the characterization after every 20 time periods based on previous 40 values.

In a similar fashion, we form features for spatial characterization. The first step is to establish vicinities of nodes that historically have been giving consistent information. Furthermore, since an agent resides on a node, all the nodes from its vicinity lie in its radio range. In this way, an *n*-gram for spatial characterization in a moment of time is made of the sensor outputs from that very moment. For example, if sensors S1, S2, S3 that belong to the same group each give the following output: 1 1 1 0 during four time epochs, we characterize them with the following set of *n*-grams (each *n*-gram contains at the first position the value of S1, the value of S2 at the second and the value of S3 at the third at a certain time epoch): 111 – occurs 3 times, 000 – occurs once, thus the feature value of each n-gram is: 111 – 0.75, 000 – 0.25, i.e. the frequencies within the observed period of time.

The same principle is followed for characterizing routes that a node has been using to send its sensed data to the sink. Each routing hop adds its ID to the message that is further forwarded, so the sink has the information about the routing path together with the message. Each sensor has its own model and each feature, i.e. n-gram in the model consists of a predefined number of successive hops used in routing information coming from the node. For example, if during the characterization time, the node has used the following paths for routing its data to the sink: A-B-C-S – 3 times, A-D-E-F-S – 2 times, A-B-E-F-S – 1 time (A – the node that is sending the data, B, C, … - other nodes in the network, S- sink), we can characterize the routing with the following n-grams (n=3): ABC, BCS, ADE, DEF, EFS, ABE and BEF. In all of the routes, the *n*-gram ABC occurs 3 times, BCS – 3, ADE – 2, DEF - 2, EFS – 3, ABE – 1, BEF – 1. The total number of *n*-grams is 15, so dividing the values given above with 15, we get the frequencies of each *n*-gram which are the values that we assign to our features, i.e. *n*-grams.

3.3.1 Deployed Distance Function

Since some of the n-grams can appear more than once, it is obvious that the extracted vectors will not be of constant size. Thus, we cannot use standard distance functions. The distance between the instances of the presented model is taken from [15]. It is designed to calculate the distance between two sequences. We have elected this one (among all given in [15]) since it is proven to be the most efficient in the terms of the absolute execution time.

The deployed distance function is actually equivalent to Manhattan distance after making the following assumption: the feature that does not exist in the first vector while exists in the second (and vice versa) actually exists with the value equal to 0, since we can say that it occurs with 0 frequency. In this way, we get two vectors of the same size and the distance between the centre and an input is between 0 (the vectors have the same features with the same feature values) and 2 (the vectors have different features with the values greater than 0). In the same way, if the set of the features of one is the subset of the feature set of the other, the distance will be between 0 and 1.

3.4 Scope of the Attacks Possible to Detect with the Approach

As previously mentioned we treat attacks as data outliers and deploy clustering techniques for attack detection. In this work we will present the unsupervised GA designed to this end. There are two possible approaches for detecting outliers using clustering techniques [16] depending on the following two possibilities: detecting outlying clusters or detecting outlying data that belong to non-outlying clusters. For the first case, we calculate the average distance of each node to the rest of the nodes (or its closest neighborhood) (*MD*). In the latter case, we calculate quantization error (*QE*) of each input as the distance from its group center.

The attacks that can be detected with the proposed approach are those that introduce changes into either the sensed value that is forwarded to the base station or the routing paths. These changes will result in different distribution of the extracted *n*-grams. However, if we take frequencies as feature values, the sum of the feature values remain the same, i.e. 1, so the following is valid:

$$\sum_{i=0}^{N} \Delta f_i = 0 \qquad (1)$$

where N is the total number of the extracted *n*-grams and Δf_i is the change of the feature value of the *n*-gram *i*. On the other hand, according to the distance function, the introduced change in distance between the attacked instance and any other is:

$$\Delta D = \sum_{i=1}^{N} |\Delta f_i| \qquad (2)$$

In essence, this is the change introduced in the above defined *QE* or/and *MD* values. Thus, the following inequality defines the changes introduced by the attacks:

$$\sum_{i=1}^{N} |\Delta f_i| > f_{th} \qquad (3)$$

where f_{th} is the threshold value used to distinguish attacks from normal situations.

Now we will see how the changes introduced by the attacker affect on the feature values. Having in mind that each sensed value or a routing hop participates in n features, where n is the size of the *n*-gram, if the attacker changes one value, the values of *2n* (at most) features will be changed, as the values of n new *n*-grams created with the change will increase, while the values of those that existed before the change will decrease.

For these reasons, if the attacker introduces N_{err} change in the sample of the size N_{sample}, the value of ΔD will range between 0 (in the case the changes are symmetric, so the effect of one change cancels the effect of another and the distribution does not change at the end), and the value that corresponds to the case when the effects of each change sum given with the following formula:

$$2n * f_{err} = \frac{2nN_{err}}{N_{sample}} \quad (4)$$

Having in mind the correlation of the n-grams, this value has to be pondered with $(1 - e^{\alpha})$, where $\alpha = 1 - 1/\rho$ and ρ is the coefficient of total correlation between the n-grams. Finally, we get the following formula:

$$(1 - e^{\alpha})\frac{2nN_{err}}{N_{sample}} > f_{th} \quad (5)$$

We take f_{th} to be 1 for the following reasons. Having in mind that the attacks will often result in creating new *n*-grams, it is reasonable to assume that the extracted vector in the presence of attackers will not be a subset of any vector extracted in normal situation, thus the distance will never be lower than 1.

Finally, this gives us the minimal number of changes the attacker has to introduce in order to be detected by the approach:

$$N_{err\min} > \frac{N_{sample}}{2n(1 - e^{\alpha})} \quad (6)$$

From this equation we can conclude that smaller characterization period (N_{sample}) and the higher *n* gives us the opportunity to detect the attacker even if he introduces small number of changes. On the other hand, this can also result in higher number of false positives, so a tradeoff between higher detection and lower false positive rate has to be established. In essence, this depends on many factors, such as the application of the deployed WSN or the existing redundancy.

3.5 Recovery from Attacks

In this work the reputation is calculated in the following way. We define two reputation values, *repQE* and *repMD* based on the previously defined *QE* and *MD* values and afterwards joint reputation *rep* used for updating overall reputation based on these two values:

```
if (QE<1) repQE = 1;        if (MD<1) repMD = 1;
else repQE=1-QE/2;            else repMD=1-MD/2;
```

For the reasons explained in the previous chapter, the value (rep) for updating overall reputation is calculated in the following way:

```
if (QE>1)rep=repQE;
else rep=repMD;
```

There are two functions for updating the overall reputation of the node, depending whether the current reputation is below or above the established threshold that distinguishes normal and anomalous behavior. If the current reputation is above the threshold and the node starts behaving suspiciously, its reputation will fall quickly. On the other hand, if the reputation is lower than the established threshold, and the node starts behaving properly, it will need to behave properly for some time until it reaches the threshold in order to "redeem" itself. In order to achieve this, we use the function x+log(1.2*x) because it provides what we want to accomplish: if x is higher than 0.5, the output rises quickly, so the reputation rises; if x is around 0.5, the output is around 0, so the reputation will not change its value significantly; if x is smaller than 0.4, the output falls below 0. Finally, the reputation is updated in the following way:

```
if (last_reputation[node]>threshold)
new_reputation[node]=last_reputation[node]+rep+log(
1.2*rep);
else
new_reputation[node]=last_reputation[node]+0.05*(re
p+log(1.2*rep));
```

If the final value falls out from the [0, 1] range, it is rounded to 0 if it is lower than 0 or to 1 in the opposite case.

The coefficient 0.05 in the second case is added with the idea of limiting the reputation increase of malicious nodes. In this way we provide a protection from selective behavior, i.e. the node has to behave correctly during the majority of time, in this case 95% of the time. In general, the value of this coefficient is a tradeoff between the false alarm rate and the amount of the time the nodes have to behave correctly, and should be adapted according to the specificities of each situation.

If during the testing of temporal coherence, we get normal data different from those that the clustering algorithms saw during the training, it is possible to get high *QE* value as well. On the other hand, the spatial coherence should not detect any anomalies. Thus, the final reputation will fall only if both spatial and temporal algorithms detect anomalies. In this way we also decrease the number of false positives. This is implemented in the following way:

```
if (value_rep <  threshold)
{     if ( space_rep <  threshold)
      result = value_rep;
    else result = 1 - value_rep; }
 else result = value_rep;
```

where `value_rep` is the reputation assigned by the algorithms for temporal characterization and `space_rep` is the reputation assigned by the algorithms for spatial characterization.

Concerning the detection of routing protocol anomalies, the explained approach can tell us if there is something suspicious in routing paths of a certain node. Yet, in order to find out the nodes that are the origin of the attack, we need to add one more step. In this step, if the reputation of the routes calculated in the previous step is lower then the established threshold, the hops that participated in the bad routes will be added to the global list of bad nodes, or if they already exist, the number of their appearance in bad routes is increased. The similar principle is performed for the correct nodes. For each node, let the number of its appearances in bad routes be *nBad* and the number of its appearances in good routes be *nGood*. Finally, if *nGood* is greater than *nBad*, the node keeps its reputation value, and in the opposite case, it is assigned the following reputation value: *nGood / (nGood + nBad)*. In this way, as the bad node spreads its malicious behavior, its reputation will gradually decrease.

3.6 Distributed Organization of Detectors

Distributed system is organized as a group of detectors, i.e. agents that execute clustering algorithms and assign reputation to sensors. Considering that there is a possibility that the attacker that has taken over a node can disable or compromise the agent that resides on that node, we introduce agent redundancy: at least three different agents will examine the behavior of each node and all will affect on its reputation. The final reputation and the final decision on a node can be implemented in various ways, such as majority voting, average reputation, weighted average reputation, etc.

The distributed approach offers numerous advantages over the base-station implementation: it provides better scalability and it does not suffer from the single point of failure problem. The detector redundancy is beneficial for two reasons. On the one hand, learning algorithms have many parameters that should be set from the beginning, e.g. number of clusters, duration of training, etc. On the other hand, this protects us in the cases when an agent resides on a compromised node, when it is possible for the attacker to compromise the agent as well, or to launch attacks against the learning system itself, such as poisoning the training data. The detector redundancy increases the efforts of the attacker necessary to compromise the detection process, as for each node the attacker first has to discover which detectors examine each node, which is something that only the base station knows, and compromise the majority of them at the same time, as in the opposite case each of the compromised agents would be discarded.

3.7 Incorporation of the Proposed Model into Envisioned WSN Model

There are various possibilities of incorporation of our detection system in the proposed WSN model (Sec. 3.1): to train the agents in either the base station or the PDA-like sensors and already trained agents that do not consume many resources are further distributed to all the nodes. Another possibility is to perform both training and detection of intrusions in the entities that are supposed to have enough resources to carry out these operations, i.e. PDA-like sensors and the base station. In this way the rest of the sensors are not affected by the incorporation of our system.

4. GENETIC ALGORITHM AS DEPLOYED

In this work an unsupervised version of genetic algorithm [17] is designed. The principal idea is to find optimal clustering, so the chromosome and the genetic operators are designed for efficient solving of this problem. In the following we will present the problem-specific GA aspects of our solution.

4.1 Chromosome Codification

A chromosome is a potential solution to clustering problem, so each gene represents a group centre. Since the optimal number of groups in a clustering problem is not known a priori, the chromosomes are implemented as lists of variable size. Each centre is presented as a collection of also variable size whose elements are the vectors of *n*-grams defined above with their corresponding feature value.

For example, if we are dealing with the sensor network for detecting binary events, the output of each sensor will have the following temporal form: 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 ..., where 1 means that the event has occurred and 0 that it has not occurred. Thus, we can extract the following set of *n*-grams: 000, 001, 011, 111, 110, 100. Since each gene represents a cluster, i.e. its centre, it is composed of a set of the *n*-grams with their corresponding frequencies. Some gene examples are given in Fig. 2. A set of the genes makes a chromosome.

4.2 Initialization Process

The input to the algorithm is the set of *n*-grams extracted as defined in the previous chapter. Their frequencies, i.e. feature values, are extracted within a predefined period of time.

All the *n*-grams extracted in the same time window with their corresponding frequencies form a collection. These collections will

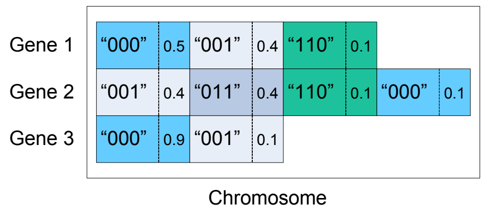

Figure 2. Chromosome Example

be the initial genes of the chromosomes. In this way, from the training set we get L_{Cin} number of collections. In the initialization process, we define the length of each chromosome as a random number L_{ch} less or equal to L_{Cin}/k, where k is a number bigger than 5 for large training sets and less than 5 for small datasets. Each gene is one of the initial collections and all genes are different (if there are at least L_{ch} different collections).

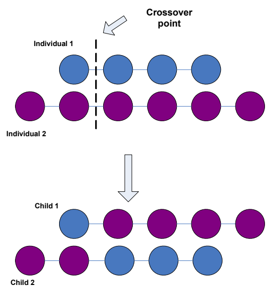

Figure 3. Crossover operator

4.3 Deployed Genetic Operators

Standard one-point crossover is deployed in this work. In this case the chromosomes are the lists of different lengths, so one-point crossover consists in selecting random crossover points in both individuals that do not necessarily have to be same, in the way depicted in Fig.3.

For the purpose of this work, we design custom mutation operator. Each gene in a chromosome is changed with a fixed low probability p_{mut} in the following way: each feature value v becomes either v*(1+2*d) or v*(1-2*d) with the same probability, where d is a random number from the range [0; 0.05]. Since v is the frequency, it can take only the values between 0 and 1, so new values will be rounded to 0 or 1 if the limits get exceeded.

4.4 Fitness Function

Bearing in mind that optimal clustering is not known a priori, it imposes the usage of a clustering validation coefficient as fitness function. The Davies-Bouldin (DB) index [18] is selected because of the following advantages over other measures:

1. Stability of results: this index is less sensitive to the position of a small group of data set members (i.e. outliers) than other measures, such as for example, the Dunn's index [18].

2. In the case of more than 2 clusters and the need to rank them, some measures (for example the Silhouette index [18]) behave unpredictably, whereas the expected behavior of the DB index in these cases is appropriate.

DB index is a function of the ratio of the sum of *within-cluster scatter* to *between-cluster separation*. The scatter within the *i*th cluster is computed as:

$$S_i = \sqrt{\frac{1}{|C_i|}\sum_{x \in C_i} \|x - c_i\|^2} \quad (8)$$

where C_i and $|C_i|$ represent the ith cluster and the number of the elements that belong to the ith cluster respectively. $\|x-ci\|^2$ is the distance between the centre and an element from the same cluster. Then, we define:

$$R_i = \max_{j, j \neq i} \left\{ \frac{S_i + S_j}{d_{ij}} \right\} \quad (9)$$

where d_{ij} is the distance between the corresponding cluster centres calculated according to the same distance function. The Davies-Bouldin *DB* index is then computed as:

$$DB = \frac{1}{K}\sum_{i=1}^{K} R_i \quad (10)$$

where K is the number of different clusters. From the formula (9) we can conclude that minimal R_i values are in the cases with the lowest within-cluster scatter and the highest between-cluster separation, which is considered as good clustering. Thus, the idea is to minimize the maximal R_i values. Consequently, the final objective is to minimize the *DB* index in order to achieve optimal clustering since small *DB* values correspond to the groups that are compact (low within-cluster scatter) and whose distances between the centers are high (high between-cluster separation). Therefore, the fitness of chromosome *j* is defined as ($1/DB_j$), where DB_j is the Davies-Bouldin index computed for this chromosome. The maximization of the fitness function will ensure minimization of the *DB* index.

5. EXPERIMENTAL EVALUATION

5.1 Detection of Attacks

The proposed algorithm has been tested on a simulator of sensor networks developed by our research group and designed using the C++ programming language. GA uses overlapping populations, where the worst 40% of individuals are exchanged in each generation. In the following experiments GA has 20 individuals, evolved during 50 generations, with respective crossover and mutation probabilities 0.6 and 0.05. Selection is performed using standard roulette wheel approach. The computational time of GA under these settings is measured in minutes.

In the following experiments we will present the performance of the approach in the presence of a representative attack on each of the core protocols. The presented results are average cases. There will be two typical situations: in the first case the attack will start after the end of training, so the training will be performed with "clean" data, while in the second case we will have the situations where the training data contains the traces of attacks as well. The aim of these experiments is to show that no constraints on training data exist.

In the first experiment we have a Sybil attack, which is one of the most aggressive and elusive attacks. In this attack, a malicious node possesses a number of valid IDs, either fabricated or stolen ones, which can be used to compromise various aspects of network functioning. In the first case, the attack starts at tick 650, while the training ends at tick 600. The reputation evolution in time is presented in Fig. 4, while the corresponding detection evolution is given in Fig. 5. We can observe in Fig.5 that both the malicious node situated at position 1247, as well as the nodes whose IDs have been stolen have their reputation lowered to 0. The false negative line that lowered to 0 demonstrates that the attack has been completely confined, i.e. that all the nodes whose IDs have been stolen are isolated from the network.

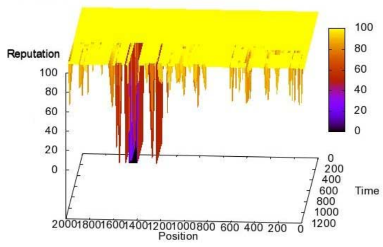

Figure 4. Reputation Evolution – Sybil attack starts at 650

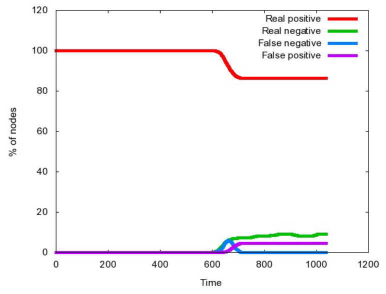

Figure 5. Detection Evolution – Sybil attack starts at 650

In the second experiment the Sybil attack starts at tick 100, while the training ends at tick 460, which means that the normal data makes 22% of the training data. This is the minimal observed rate that still permits detection and confinement of the attack. The results are presented in Figs. 6 and 7. Again, we can clearly distinguish low reputation of the malicious nodes and the nodes whose IDs have been stolen. False negative rate equal to 0 proves that the attack is detected and completely confined. Although the nodes whose ID has been stolen by the malicious nodes are not necessarily malicious, they have been compromised, and thus have to be isolated. The base station can deploy further measures, such as assign new secret keys and change their reputation.

In the following experiments we will present the results in the presence of wormhole attack, which is an attack on the routing protocol. This attack results in a connection between the nodes that are more than one hop away from each other. In the first experiment the attack starts at tick 650, while the training ends at 600. The reputation evolution is given in Fig. 8. The wormhole was launched from the position 12, and we can see that its reputation is lowered to 0. Thus, the attack is detected and completely confined. However, we can also observe that there are two more nodes with their reputation lowered, one to 21 and the other to 23. The decision whether they are compromised or not will depend on the elected threshold value. If this value is 20 or lower, we will have 0% false positive rate, but if the threshold is higher, we will have the false positive rate of 2%.

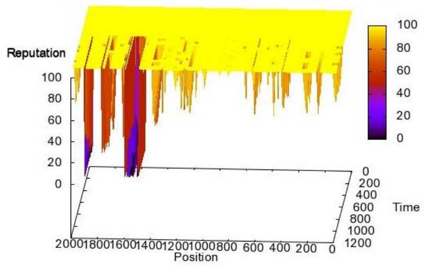

Figure 6. Reputation Evolution – Sybil attack starts at 100

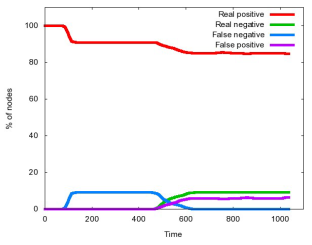

Figure 7. Detection Evolution – Sybil attack starts at 100

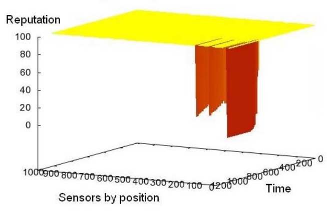

Figure 8. Reputation Evolution – Wormhole starts at 650

In the following experiment the wormhole starts at tick 250, while the training ends at 600. The results are presented in Fig. 9. We can see that the node of the approximate position 1000 (the exact position is 954) is lowered to 0. This node is the origin of the attack. Few nodes around position 280 have also low reputation. This is probably due to the proximity to the link node in the wormhole which is at position 283 (its reputation is lowered to 10). The rest of the nodes with lowered reputation and their reputations are the following: nodes at positions 279, 278, 275 with their respective reputations 7, 18 and 16. Due to their low reputation, in most of the cases they would be reported as compromised. The node at the position 219 also has lowered reputation up to 47, but its possibility to be reported as compromised is lower than for the previous nodes.

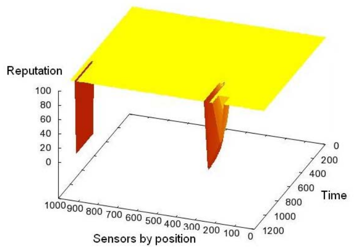

Figure 9. Reputation Evolution – Wormhole starts at 250

Finally, we present the performance in the presence of a pulse-delay attack, which introduces latencies. The result of this can be that the received critical information is not up to date. In the first experiment the attack starts at tick 650, while the training ends at 560. The attacked node is situated at position 1598. The results are presented in Fig. 10. As we can observe, the only node with lowered reputation is the compromised one. Thus, the attack is detected and confined with no false positives.

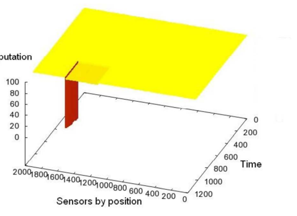

Figure 10. Reputation Evolution – Pulse-delay starts at 650

In the next experiment the attack starts at tick 400, while the end of the training remains the same. The attacked node is situated at position 793. As we can observe in Fig.11, the only node with lowered reputation is the compromised one. Thus, one more time the attack is detected and confined with no false positives.

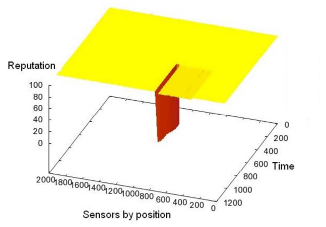

Figure 11. Reputation Evolution – Pulse-delay starts at 400

5.2 Network Survivability

We say the correct operation is maintained while the network provides the correct picture of the observed phenomenon during the whole time, although the attack has not been completely isolated. In the most general case, the attacker has to compromise at least $\lceil N/2 \rceil$ sensors, where N is the number of sensors in the given network. In the following experiment the isolated nodes do not participate in the process of making decision, thus the final decision is made based on majority decision of the rest of the nodes.

The next experiment is performed in the same surrounding as the previous ones, in the presence of two attacks, the Sybil attack and the chain attack, which consists of *n* compromised nodes, where the first (*n-1*) always forward the data to the next one in the chain, while the last one performs misrouting. The attacks compromise random nodes and it is assumed that the nodes from the whole network can be compromised. In Fig. 12 the dependence of the total amount of compromised nodes (as % of total number of nodes) on the amount of clean data (i.e. the data without traces of attacks) during the training is presented. We observe that in the presence of a detection mechanism the attacker has to introduce more effort in order to compromise the network, where the maximal percentage of compromised nodes that permits network survivability is 83.5%in the case of the Sybil, and 90% in the case of the chain attack, when the attack in both cases starts after the end of training. As the percentage of clean data decreases, it is harder to detect all the malicious nodes, for which the attacker needs to introduce less effort in order to compromise the network. However, in the case of the chain attack the effort is always much higher than in the case there is no detection mechanism. On the other hand, in the case of the Sybil for the situations the training data contains at least 10% of the clean data, the presence of a detection mechanism increases the effort of the attacker necessary to compromise the network.

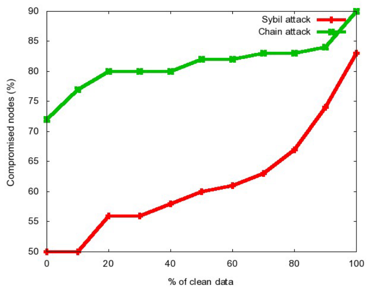

Figure 12. Max. % of compromised nodes

5.3 Resource Consumption

With the aim of proving the viability of performing the training in a PDA-like device, we have carried out the evaluation of the resource consumption using a SONY Ericsson XPERIA X10 Mini with Qualcomm MSM7227 600MHz CPU and Android OS 1.6. This is an average PDA.

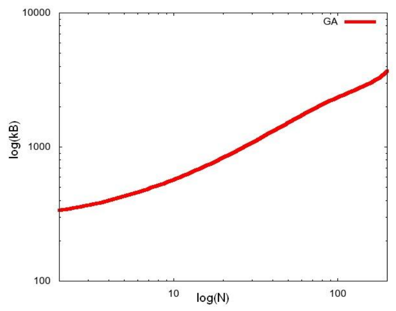

Figure 13. GA Memory Consumption vs. Number of Nodes

Regarding memory consumption, we have performed the experiments varying the number of nodes that are being examined by one PDA. The results are presented in Fig. 13 varying the number of nodes from 2 to 200. The corresponding memory consumption spans from 336kB to 3670kB, which is viable for a PDA implementation. However, it is still too high to be implemented in ordinary sensor nodes. Regarding the energy consumption, in the case of 40 nodes the full battery provides enough energy for executing around 1 million training periods.

6. CONCLUSIONS

In this work we have presented an enhancement of reputation systems for WSNs that consists in adding an unsupervised GA, with the aim of detecting early traces of attacks and thus providing fast reaction to thwart activities. It has been demonstrated that the approach is capable of detecting and confining representative attacks on core network protocols with low false positive rate, without any constraint on the training data. Furthermore, it has been demonstrated that the presence of detection algorithms increases the efforts the attacker has to introduce in order to compromise the network. The viability of the approach using PDA-like devices is also proven.

7. ACKNOWLEDGMENTS

This work was funded by the Spanish Ministry of Industry, Tourism and Trade, under Research Grant TSI-020301-2009-18 (eCID), the Spanish Ministry of Science and Innovation, under Research Grant TEC2009-14595-C02-01, and the CENIT Project Segur@.

8. REFERENCES

[1] Mármol, F. and Pérez, G. Security threats scenarios in trust and reputation models for distributed systems, Computers & Security 28 (2009) 545-556.

[2] Aliguliyev, R. M. Performance evaluation of density-based clustering methods, *Inf. Sciences* 179 (2009) 3583 - 3602.

[3] Moya, J. M. et al. Improving security for SCADA sensor networks with reputation systems and Self-Organizing maps, *Sensors* 9 (2009) 9380-9397.

[4] Boukerch, A.; Xu, L.; EL-Khatib, K. Trust-based Security for Wireless Ad Hoc and Sensor Networks. *Comput. Commun*. **2007**, *30*, 2413–2427.

[5] Papaioannou, T.G.; Stamoulis, G.D. Effective use of reputation of peer-to-peer environments. In *Proc. of IEEE/ACM CCGRID 2004, GP2PC workshop.* IEEE Comp. Soc: Chicago, IL, USA, April 19-22, 2004; pp. 259-268.

[6] Antoniadis, P.; Courcoubetis, C.; Efstathiou, E.; Polyzos, G.; Strulo, B. Peer-to-Peer wireless LAN consortia: Economic modeling and architecture. In *Proc. of the Third IEEE Int.l Conf. on Peer-to-Peer Comp.,* IEEE Comp. Soc.: Linköping, Sweeden, Sept. 1-3, 2003**;** pp. 198-199.

[7] Hu Y, Perrig A, Johnson D. Packet leashes: a defense against wormhole attacks in wireless networks. Proc. of 22nd annual joint conference of the IEEE computer and communications societies (INFOCOM 2003). vol. 3, 2003. pp. 1976–86.

[8] Bar El, H. Introduction to Side Channel Attacks. Discretix Technologies Ltd., 2003.

[9] Mukhopadhyay D, Saha I. Location verification based defense against sybil attack in sensor networks. In: Distr. comp. and networking. Springer, 2006. p. 509–521.

[10] Liu, Y.G., Chen, K.F., Liao, X.F., Zhang ,W. A genetic clustering method for intrusion detection. Pattern Recognition, 37(5), 927-942. (2004)

[11] Renjit, J. A. and K. L. Shunmuganathan, Distributed and cooperative multi-agent based intrusion detection system, Indian Journal of Science and Technology 3(10): 1070-1074. (2010).

[12] Hortos, W. S. Unsupervised algorithms for intrusion detection and identification in wireless ad hoc sensor networks, Proc. SPIE 7352, pp. 73520J, 2009.

[13] Renjit, J. A. and K. L. Shunmuganathan. Distributed and cooperative multi-agent based intrusion detection system, Indian Journal of Science and Technology 3(10): 1070-1074. (2010).

[14] Roosta, T. G. Attacks and Defenses on Ubiquitous Sensor Networks, Ph. D. Dissertation, Univ.of California at Berkeley, 2008

[15] Rieck, K.; Laskov, P. Linear Time Computation of Similarity for Sequential Data. *J. Mach. Learn. Res.* **2008,** *9*, 23-48.

[16] Muñoz, A.; Muruzábal. J. Self-Organizing Maps for Outlier Detection. *Neurocomputing 18(1-3),* **1998**, 33-60

[17] Goldberg, D. E. *Genetic algorithms for search, optimization, and machine learning*. 1st Ed. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.

[18] Mitra, S.; Acharya, T. *Data Mining: Multimedia, Soft Computing, and Bioinformatics;* John Wiley & Sons, Inc.: Hoboken, New Jersey, USA, 2003; pp. 257-269