Detection of Eavesdropping Attack in UAV-aided Wireless Systems: Unsupervised Learning with One-Class SVM and K-means Clustering

Detection of Eavesdropping Attack in UAV-aided Wireless Systems: Unsupervised Learning with One-Class SVM and K-means Clustering

Tiep M. Hoang, Nghia M. Nguyen, and Trung Q. Duong

Abstract—A wireless relaying system in the presence of an active eavesdropper is considered. The system is assumed to use an unmanned aerial vehicle (UAV) as a relay and the uplink phase is for authentication. The active eavesdropper is assumed to sneak into the system by attacking the UAV during the process of authentication. To detect the eavesdropping attacks, we consider building predictive models with one-class support vector machines (OC-SVM) and K-means clustering. To prepare datasets for training predictive models, we propose a framework for creating the features of testing data from wireless signals and another framework for generating training data. Our results show the superiority of OC-SVM over K-means in terms of stability, while K-means clustering is better when the eavesdropper uses high power in transmission.


I. INTRODUCTION

The application of machine learning (ML) to physical layer security (PLS) in wireless systems (WSs) is, in general, challenging and widely open. On one hand, the feasibility of many ML algorithms has not been investigated yet. It is also unclear whether an algorithm working well on a dataset will work well on others. On the other hand, there is no standard for forming structured data from wireless signals. Due to the fact that ML algorithms learn from data, there is a need to build up the data from wireless signals in WSs. Motivated by the above-mentioned, we aim to use unsupervised learning for PLS in WSs. Instead of considering general WSs, we focus on WSs based on unmanned aerial vehicles (UAVs) since UAV-aided WSs are now emerging as part of 5G [1].

Most previous works on UAV-aided WSs focused on measuring secure performance through secrecy rate and/or secrecy outage probability (see [2] and [3]), but did not consider the detection of eavesdropping attacks. In [4], the PLS aspect of a UAV-aided WS is investigated with the emphasis on authentication. However, the detection method used in [4] relies on a log-likelihood radio but not ML techniques. Just recently, [5] has suggested using reinforcement learning to deal with eavesdroppers in UAV-aided WSs. Except [4] and [5], it can hardly find any closely-related work. Although there are other works using ML for intrusion detection in WSs, e.g., [6] and [7], those works study security at the upper layers rather than the physical layer. In short, from a PLS-related viewpoint, the applicability of ML for PLS in WSs (as well as in UAV-aided WSs) is vague and needs to be clarified.

In this work, we study the feasibility of applying ML to detect active eavesdropping in UAV-aided WSs. Different from [5] which concentrates on reinforcement learning, we consider unsupervised learning. Motivated by the facts that i) there is no single unsupervised learning method to be appropriate for all datasets and that ii) a dataset should be checked with several unsupervised learning methods to determine the most appropriate one for usage, we respectively propose using one-class support vector machines (OC-SVM) and K-means clustering for the detection of attacks. To the best of our knowledge, there is not any previous work making use of these two methods for UAV-aided WSs. Moreover, we present how to create testing data from wireless signals before entering the data into ML models. We also present how to create artificial training data (ATD) based on the statistical knowledge of channel state information (CSI). While the ATD is used for training OC-SVM models, it is only used for labeling the clusters of K-means models. The strength and frequency of attacks, which are reflected through the eavesdropper’s power and the ratio of 2 labels, are considered. Through numerical results, the advantages of OC-SVM and K-means are compared in terms of accuracy, sensitivity and specificity.

II. SYSTEM MODEL

Assume that a time slot includes two equal phases. In phase 1, Bob and Eve transmit signals to Drone for the purpose of requesting messages. In phase 2, Drone amplifies and forwards its received signals to Alice for anomaly detection.

A. Phase 1: Bob and Eve send pilot vectors to Drone

In the i-th phase of the i-th time slot, Bob transmits $s_B^{[i,1]}$ to Drone over the distance $d_{BD}d_{ef}$, while Eve transmits $s_E^{[i,1]}$ to Drone over the distance $d_{ED}d_{ef}$. Herein, $d_{ef}$ is a reference distance. We assume that $E_{s_B^{[i,1]}}\left\{\left|s_B^{[i,1]}\right|^2\right\} = P_B$ and $E_{s_E^{[i,1]}}\left\{\left|s_E^{[i,1]}\right|^2\right\} = P_E$. The large-scale fading of the Bob-Drone link is represented by $\xi d_{BD}^{\alpha_L}$, while that of the Eve-Drone link is represented by $\xi d_{ED}^{\alpha_L}$. Herein, $\alpha_L$ is the path-loss exponent. Meanwhile, $\xi$ can be calculated as $\xi = (4\pi f/S_{\text{light}})^{-\alpha_L} 10^{-L_{\text{average}}/10}$ in which $f = 800$ MHz.
is the carrier frequency, \( S_{\text{light}} \approx 3 \times 10^8 \text{ m/s}^2 \) is the speed of light, and the average value of additional loss due to LoS link \( L_{\text{average}} \) is set to 2 dB. Regarding the small-scale fading, it is shown that Rician fading can be used in UAV-aided wireless systems [8], [9]. Denote \( h_{BD} \) and \( h_{ED} \) be the small-scale fading of the Bob-Drone link and that of the Eve-Drone link, respectively. We can express \( h_{BD} \) as

\[
h_{BD} = \sqrt{K_{BD}/(K_{BD} + 1)} h_{h,\text{dB}} + \sqrt{1/(K_{BD} + 1)} h_{N,\text{dB}} \tag{1}
\]

where \( K_{BD} \gg 1 \) (say, \( K_{BD} \to \infty \)) is the Rician K-factor, \( h_{h,\text{dB}} = \mathbb{E}\{ |h_{BD}|^2 \} \) is a deterministic value, and \( h_{N,\text{dB}} \sim \mathcal{CN} (0, \mathbb{E}\{ |h_{BD}|^2 \} ) \) is a circularly symmetric Gaussian random variable (RV). Note that \( h_{BD} \) also takes the form similar to (1) and is not presented due to the space limitation.

As such, the received signal at Drone during the 1-st half of time slot \( i \) can be expressed as

\[
r_{D}^{[i,1]} = n_{D}^{[i,1]} + \sqrt{\xi d_{BD}^{-\alpha_L}} h_{BD}^{[i,1]} s_{B}^{[i,1]} + \sqrt{\xi d_{ED}^{-\alpha_L}} h_{ED} s_{E}^{[i,1]} \tag{2}
\]

where \( n_{D}^{[i,1]} \sim \mathcal{CN} (0, \sigma_{D}^2) \) be the additional Gaussian white noise (AGWN) at Drone. The drone has a limited-capacity information processor and thus, it cannot detect eavesdropping attacks. It is practical to consider that Drone only forwards \( r_{D}^{[i,1]} \) to Alice and leaves the detection task for her.

**B. Phase 2: Drone sends its signals to Ground Control Station**

In the 2-nd phase of the \( i \)-th time slot, Drone amplifies and forwards received signals to Alice. Obviously, Drone leaves the detection task for Alice. Drone contacts Alice frequently and thus, they have the statistical knowledge of the CSI of the D-A channel. The signal transmitted by Drone can be designed as

\[
s_{D}^{[i,2]} = c_{D} w_{D} r_{D}^{[i,1]} \tag{3}
\]

where \( w_{D} \) is designed based on the D-A channel such that \( \mathbb{E}\{ |w_{D}|^2 \} = 1 \), and the constant \( c_{D} \) is to guarantee that the mean power of \( s_{D}^{[i,2]} \) is equal to the transmit power \( P_{D} \). If we design \( w_{D} = (h_{DA})^*/|h_{DA}| \), then we have \( c_{D} = \sqrt{P_{D}/\mathbb{E}}\) where \( \mathbb{E} \triangleq \mathbb{E}\{ |r_{D}^{[i,1]}|^2 \} = \sigma_{D}^2 + \xi d_{BD}^{-\alpha_L} \mathbb{E}\{ |h_{BD}|^2 \} P_{B} + \xi d_{ED}^{-\alpha_L} \mathbb{E}\{ |h_{ED}|^2 \} P_{E} \) is a constant. When \( s_{D}^{[i,2]} \) is transmitted from Drone to Alice over the distance \( d_{DA} d_{ref} \), the received signal at Alice can be given by

\[
r_{A}^{[i,2]} = n_{A}^{[i,2]} + r_{A}^{[i,2]} = n_{A}^{[i,2]} + \sqrt{\xi d_{DA}^{-\alpha_L}} h_{DA} s_{D}^{[i,2]} \tag{4}
\]

where \( n_{A}^{[i,2]} \sim \mathcal{CN} (0, \sigma_{A}^2) \) is the AGWN at Alice, \( h_{DA} \) presents the small-scale fading and can be formulated in the same manner as (1). We substitute (3) into (4) to attain

\[
r_{A}^{[i,2]} = n_{A}^{[i,2]} + \sqrt{(P_{D} \xi d_{DA}^{-\alpha_L}/\mathbb{E}) |h_{DA}|} \times \left( n_{D}^{[i,1]} + \sqrt{\xi d_{BD}^{-\alpha_L} h_{BD} s_{B}^{[i,1]} + \xi d_{ED}^{-\alpha_L} h_{ED} s_{E}^{[i,1]} } \right). \tag{5}
\]

**Remark 1.** Eve designs \( s_{E}^{[i,1]} = s_{B}^{[i,1]} \) to deceive Alice into thinking that two legal users require the same message. As a result, in the downlink Alice will send the message, which is inherently intended for Bob, to both Bob and Eve. Note that this work does not focus on the transmission from Alice to Bob and Eve. Instead, this work focuses on how to know whether Eve deceives Alice or not. The only information Alice can use is the received signal \( r_{A}^{[i,2]} \) and she has to find way to detect the Eve’s presence.

Apart from \( r_{A}^{[i,2]} \), Alice does not have other information, such as the direction of arrival and frequency asynchronism [10]. Although Alice can benefit from those metrics, having so much information is ideal and out of the scope of this paper.

**III. FEATURE EXTRACTION**

Using sampling techniques, Alice can obtain realizations of \( r_{A}^{[i,2]} \) at different instants. With these samples, Alice can create structured datasets, which are inherently unavailable. This section, thus, presents how to form structured datasets, which will be fed into ML algorithms for classification tasks.

**A. Testing Data**

Suppose that in a time slot Alice receives \( N \) instantaneous values of \( r_{A}^{[i,2]} \). Alice can build up the sequence \( S_{[i]} = \{ r_{A}^{[i,2]}(1), r_{A}^{[i,2]}(2), \ldots, r_{A}^{[i,2]}(N) \} \) and she treats it like draw data. Based on \( S_{[i]} \), Alice can extract the mean \( m_{S_{[i]}} \) and the variance \( v_{S_{[i]}} \) of the sequence, i.e.,

\[
m_{S_{[i]}} = (1/N) \sum_{n=1}^{N} r_{A}^{[i,2]}(n), \tag{6}
\]

\[
v_{S_{[i]}} = (1/N) \sum_{n=1}^{N} [r_{A}^{[i,2]}(n) - m_{S_{[i]}}]^2. \tag{7}
\]

Defining the \( i \)-th data point be \( x_{i} = (m_{S_{[i]}}, v_{S_{[i]}}) \in \mathbb{R}^2 \), we have a dataset of \( Q \) testing data points after \( Q \) time slots. From (5)–(7), one can readily see that the position of \( x_{i} \) in \( \mathbb{R}^2 \) depends on the value of \( s_{E}^{[i,1]} \). From a security perspective, we only want to know the relation between the position of \( x_{i} \) and the event \( s_{E}^{[i,1]} \neq 0 \). There are two cases:

- \( s_{E}^{[i,1]} \neq 0 \) means that Eve breaks into the system by impersonating Bob during the \( i \)-th time slot. In this case, the data point \( x_{i} \) is associated with the label +1.
- \( s_{E}^{[i,1]} = 0 \) means that there is not any attack during the \( i \)-th time slot. In this case, \( x_{i} \) is associated with the label −1.

As such, the main task of Alice is to map all data points in \( \mathcal{X} = \{ x_{1}, \ldots, x_{Q} \} \) to \( \{ \pm 1 \} \). Note that the structure of a testing dataset takes the following form:

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Labels (attack or non-attack)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_{S_{[1]}} )</td>
<td>( v_{S_{[1]}} )</td>
<td>Not yet identified</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( m_{S_{[Q]}} )</td>
<td>( v_{S_{[Q]}} )</td>
<td>Not yet identified</td>
</tr>
</tbody>
</table>

**B. Artificial Training Data with One Label**

Similar to [4], we consider the worst case where the Eve’s CSI is unknown. Let us assume that Alice has knowledge of the B-D and D-A channels. Then Alice can create ATD on her
own based on simulating the entire process of transmitting signals from Bob to Alice via Drone. Note that all ATD points are associated with the label \(-1\), because Alice creates artificial signals by simulating the case of non-eavesdropping.

The process of creating ATD can be described as follows: Alice first generates the artificial signal \(\hat{r}_{A}^{[1,2]}\) which she may be likely to receive in the case of non-eavesdropping.

\[
p_{A}^{[1,2]} = \hat{n}_{A}^{[1,2]} + \sqrt{(P_D \xi d_{DA}^{-\alpha L}/\hat{\omega})} \hat{n}_{DA} + \xi d_{BD}^{-\alpha} \hat{n}_{BD} s_{B}^{[1,2]} \tag{8}
\]

where \(\hat{\omega} = \sigma_{0}^{2} + \xi d_{BD}^{-\alpha} E \{||h_{BD}||^{2}\} P_{B}, \hat{n}_{A}^{[1,2]} = n_{A}^{[1,2]}, \hat{n}_{DA} = n_{DA}, \hat{n}_{BD} = n_{BD}\). The notation \(X_{1}^{\text{dist}} = X_{2}\) means that \(X_{1}\) has the same distribution as \(X_{2}\). Using \(\hat{r}_{A}^{[1,2]}\), Alice then computes the quantities: \(\hat{S}_{i}^{[1]} = (1/N) \sum_{n=1}^{N} \hat{r}_{A}^{[1]}[n] \) and \(\hat{v}_{S_{i}}^{[1]} = (1/N) \sum_{n=1}^{N} \hat{r}_{A}^{[2]}[n] \). Finally, Alice ends up with the ATD which is described as the following tabular fashion:

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Labels (non-attack)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\hat{m}<em>{S</em>{i}}^{[1]})</td>
<td>(\hat{v}<em>{S</em>{i}}^{[1]})</td>
<td>(-1)</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>(\hat{m}<em>{S</em>{i}}^{[Q]})</td>
<td>(\hat{v}<em>{S</em>{i}}^{[Q]})</td>
<td>(-1)</td>
</tr>
</tbody>
</table>

Herein, \(Q\) is the number of ATD points and can be determined by Alice. The ATD can be viewed as the array \(\hat{X} = \{\hat{x}_{1}, \ldots, \hat{x}_{Q}\}\) with each point \(\hat{x}_{i} = (\hat{m}_{S_{i}}, \hat{v}_{S_{i}}) \in \mathbb{R}^{2}\) being labelled as \(-1\).

IV. USING UNSUPERVISED LEARNING FOR ANOMALY DETECTION

This section presents how Alice can employ OC-SVMs and K-means to determine if the system is under attack.

A. One-Class Support Vector Machines

As a variation of the standard SVM, OC-SVM is often used to identify abnormalities with the outstanding ability of avoiding over-fitting problems [11]–[13]. In principle OC-SVMs use a mapping \(\phi(\cdot): \mathcal{X} \rightarrow \mathcal{F}\) to tretch the input space \(\mathcal{X}\) to a higher-dimensional space \(\mathcal{F}\) (namely, the feature space). OC-SVMs then create a region such that most of the newly-transformed ATD points (i.e., \(\phi(\hat{x}_{1}), \ldots, \phi(\hat{x}_{Q})\)) fall within the region, while the remaining points lie outside the region [13]. Finally, OC-SVMs transform all testing data points into \(\phi(x_{1}), \ldots, \phi(x_{Q})\) and label these new points. If any \(\phi(x_{i})\) lies outside the predetermined region, it will be considered an outlier and will take the label \(+1\). Otherwise, it will be considered an inlier and will take the label \(-1\). To illustrate the way OC-SVMs operate, we depict Fig. 1.

A testing data point \(x_{i} = (\hat{m}_{S_{i}}, \hat{v}_{S_{i}})\) (equivalently, \(\phi(x_{i})\) in the feature space) is a true outlier if it is actually associated

\[\text{with } s_{B}^{[1,2]} \neq 0. \text{Unfortunately, due to the random nature of } S_{i}, i \in \{1, \ldots, Q\}, \text{the data point of interest might be possibly misclassified when being put inside the region.}

B. K-means ++ Clustering

Totally different from OC-SVMs, K-means clustering does not require training data but works on testing data directly. In particular, K-means clustering divides the testing data points into K clusters. For binary classification (attack and non-attack), we set K to be K= 2. The drawback of K-means clustering is that the resultant clusters will not be labelled. We will overcome this difficulty by using the ATD. Herein, the use of the ATD is not for training K-means clustering models, but for labelling the resultant clusters.

In principle, \(\mathcal{X}\) is divided into two disjoint subsets each corresponding to a cluster. Denote \(\mu_{k}\) (with \(k \in \{1, 2\}\)) be the center of cluster \(k\). To find \(\{\mu_{k}\}_{k=1}^{2}\), K-means clustering performs an iterative algorithm for which the value of \(\mu_{k}\) at the \((r + 1)\)-th iteration, i.e., \(\mu_{k}^{(r+1)}\), is updated by the assignment operator \(\mu_{k} = \mu_{k}^{(r+1)}\). In the K-means algorithm, the initial position of center \(k\), i.e., \(\mu_{k}^{(0)}\), is arbitrarily chosen. To improve the accuracy for K-means, [14] suggested a seeding technique for choosing initial centers, i.e., K-means ++. An illustration of using K-means ++ is presented in Fig. 2.

Assigning labels to clusters: Let \(\mu_{ATD}\) be the center of the ATD. The coordinates of \(\mu_{ATD}\) in \(\mathbb{R}^{2}\) can be found by calculating the average of \(\hat{m}_{S_{i}}^{[1]}\) and that of \(\hat{v}_{S_{i}}^{[1]}\) over \(Q\) discrete values, i.e., we have \(\mu_{ATD} = (E_{i} \{\hat{m}_{S_{i}}^{[1]}\}, E_{i} \{\hat{v}_{S_{i}}^{[1]}\})\) where \(E_{i} \{\hat{m}_{S_{i}}^{[1]}\} = (1/Q) \sum_{i=1}^{Q} \hat{m}_{S_{i}^{[1]}}\) and \(E_{i} \{\hat{v}_{S_{i}}^{[1]}\} = (1/Q) \sum_{i=1}^{Q} \hat{v}_{S_{i}^{[1]}}\). At this stage, we are able to determine which cluster should be labelled as \(-1\). If \(|\mu_{1} - \mu_{ATD}| \leq |\mu_{2} - \mu_{ATD}|\) holds true, the 1-st cluster and its corresponding data points will be labelled as \(-1\). In contrast, if \(|\mu_{1} - \mu_{ATD}| > |\mu_{2} - \mu_{ATD}|\), the 1-st cluster and its corresponding data points will be labelled as \(+1\).
the decrease in $P$ is less than usual. For example, $P$ data points - which are associated with Bob. falls below attacks the system frequently or not, the difference in the three $P$ data points are associated with eavesdropping attacks. Table OC-SVM and K-means ++. The increase in eavesdropping attacks. By defining $P$ and presented a framework for creating relevant features from $\mu^{ATD}$ can be considered constant.

Let $Q_{(+1)}$ be the number of data points associated with eavesdropping attacks. By defining $p = Q_{(+1)}/Q$, we examine the impact of $p$ on the accuracy, sensitivity and specificity for OC-SVM and K-means ++. The increase in $p$ implies that Eve tends to perform more attacks on the system. Similarly, the decrease in $p$ implies that Eve tends to attack the system less than usual. For example, $p = 0.5$ means 50% of testing data points are associated with eavesdropping attacks. Table I shows that $p$ has a low impact on the accuracy, sensitivity and specificity for both types of ML algorithms. Whether Eve attacks the system frequently or not, the difference in the three performance metrics is not actually significant. In this paper, $p$ falls below 50% to make sure the dominance of (normal) data points - which are associated with Bob.

The results in Table I also show that the K-means ++ approach is better than the OC-SVM approach at high $P_E/\sigma_E^2$. However, at moderate and low $P_E/\sigma_E^2$, the OC-SVM algorithm is the better one. We can observe that the OC-SVM algorithm seems to be less sensitive to the change of $P_E$. On the contrary, the K-means ++ algorithm tends to reduce the secure performance quickly, e.g., from 100% to 48.8% in accuracy, when $P_E/\sigma_E^2$ decreases from 20 dB to 0 dB. It should also be noted that Eve might sneak into the system by reducing as much $P_E$ as possible in order to make Alice think of it as noise. As just discussed, the K-means ++ algorithm fails to detect eavesdropping attacks (for example, the accuracy falls below 55% at $P_E/\sigma_E^2 = 0$ dB). As such, in terms of stability, the OC-SVM approach reveals that it is more suitable than the K-means ++ counterpart. Numerically, OC-SVMs in Table I maintain the accuracy of around 80-90% in most cases.

VI. CONCLUSIONS

We considered a new flying ah-hoc network based on UAVs and presented a framework for creating relevant features from wireless signals. We also proposed creating artificial training data, which plays an important role in the training process of OC-SVMs as well as in assigning labels to the clusters of K-means ++. Through numerical results, we observed and suggested that K-means ++ should be used in the case of high $P_E/\sigma_E^2$. On the contrary, OC-SVMs should be preferable to K-means ++ when the Eve’s power $P_E$ is changeable.

REFERENCES