IntruMine: Mining Intruders in Untrustworthy Data of Cyber-physical Systems

Lu-An Tang[1], Quanquan Gu[1], Xiao Yu[1], Jiawei Han[1]
Thomas La Porta[2], Alice Leung[3], Tarek Abdelzaher [1], Lance Kaplan [4]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign;
[2]Department of Computer Science, Pennsylvania State University;
[3]BBN Technology, [4]U.S. Army Research Laboratory
{tang18,qgu3,xiaoyu1,hanj,zaher}@illinois.edu
tlp@cse.psu.edu, aleung@bbn.com, lance.m.kaplan.civ@mail.mil

Abstract

A Cyber-Physical System (CPS) integrates physical (*i.e.*, sensor) devices with cyber (*i.e.*, informational) components to form a situation-aware system that responds intelligently to dynamic changes in real-world. It has wide application to scenarios of traffic control, environment monitoring and battlefield surveillance. This study investigates the specific problem of intruder mining in CPS: With a large number of sensors deployed in a designated area, the task is real time detection of intruders who enter the area, based on untrustworthy data. We propose a method called *IntruMine* to detect and verify the intruders. *IntruMine* constructs *monitoring graphs* to model the relationships between sensors and possible intruders, and computes the position and energy of each intruder with the link information from these monitoring graphs. Finally, a confidence rating is calculated for each potential detection, reducing false positives in the results. *IntruMine* is a generalized approach. Two classical methods of intruder detection can be seen as special cases of IntruMine under certain conditions. We conduct extensive experiments to evaluate the performance of *IntruMine* on both synthetic and real datasets and the experimental results show that *IntruMine* has better effectiveness and efficiency than existing methods.

1 Introduction

A Cyber-Physical System (CPS) is an integration of sensor networks with informational devices. Examples of CPS include many promising applications, such as traffic monitoring [10, 19], environment control [17] and national border patrol [4]. In particular, CPS has important applications on the battlefield to help protect troops and civilians [18]. Such a system employs a large number of low cost, densely deployed sensors to watch over designated areas and automatically detect possible intruders [4]. Figure 1 shows the framework of a battlefield CPS. The sentry nodes are the deployed sensors. They constantly collect sound, magnetic or temperature data from the environment. Land-based gateways and aircraft bridges transmit sensor data to a command center. The system in the command center analyzes the collected data to produce information about intruders. Such technology can enable military forces to see through the "fog of war" and improve decision making.

The key task in such a scenario is to mine the real intruder information from a large set of untrustworthy data. There are four major challenges:

- Untrustworthy data: The collected data are highly unreliable due to hardware and communication limits. Many deployment experiences have shown that untrustworthy data is the most serious problem that impacts CPS performance. Buonadonna *et al.* point out that faulty data can occur in various unexpected ways and less than 69% of their data could be used for meaningful interpretation [17]. Szewczyk *et al.* also find that about 30% of data are faulty in their deployment [15]. It is difficult to filter out untrustworthy data records solely based on the data values, since most faulty records have values similar to real ones.
- Complex Requirements: Many intruder detection algorithms rely on prior knowledge of the number of intruders, movement speed, or energy (*i.e.*, the emitted signal strength) [14, 5, 2, 11]. However, the users often cannot provide such attributes of intruders in real applications. In contrast, they would like to obtain fine-grained situational awareness of

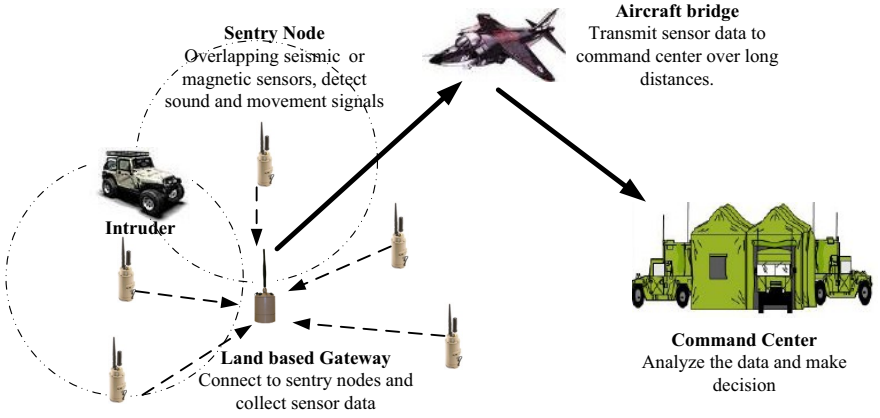

Figure 1: The Framework of a Battlefield CPS

the battlefield and require the system to generate this information automatically. For example, the threat posed by a vehicle is much higher than the threat posed by a pedestrian. Since the vehicle and pedestrian have different energy (*i.e.*, emitted signal strength), the users would like to know the intruder's energy to judge the threat level.

- Unsupervised Learning: Several intrusion detection methods build up the models or classifiers based on a training dataset [13, 12]. However, such training datasets are hard to get in realistic deployments. It is costly and error-prone to manually label a large sensor dataset. In addition, the data are based on a specific deployment plan, hence it is usually difficult to apply a training set from one deployment to another. To increase the system feasibility, the intruder mining algorithm should use unsupervised learning that does not require a large training dataset.
- Large Scale: A typical CPS includes hundreds, even thousands of sensors [1]. Each sensor generates a reading every few minutes, and the readings form a huge data stream. Furthermore, many applications require immediate action against intruders. The mining must be efficient to process the huge data stream and find intruders in real time.

In this paper, we propose a novel framework, *IntruMine*, to find real intruders from untrustworthy data. *IntruMine* is unsupervised yet effective. It iteratively models the relationship between sensors and intruders via *monitoring graphs*, and estimates the attribute values of the intruders based on the link information of such graphs. The *confidence* of intruder detection is computed based on the difference between the real sensor readings and the estimated ones. This measurement is used to verify the detected intruders and filter out false positives. The contributions of this paper can be summarized as follows.

- Proposing a monitoring graph model and conducting the intruder mining using such a graph.
- Introducing the concepts of intruder confidence to verify the detected intruders.
- Showing that *IntruMine* is well connected to two classical models, which are proved to be special cases of *IntruMine* under certain conditions.
- Carrying out extensive experiments to evaluate the effectiveness and efficiency of *IntruMine* on both synthetic and real datasets. Our experiments show that *IntruMine* yields higher precision and time efficiency than existing methods.

Since the battlefield surveillance is one of the most important application of CPS, we motivate the problem by examples of detecting intruders on battlefield. However, the proposed framework is flexible to extend to other scenarios, such as environmental monitoring, traffic control and national border patrol.

The rest of the paper is organized as follows. Section 2 introduces the preliminary knowledge and problem formulation; Section 3 proposes the intruder mining techniques; Section 4 discusses the connections between *IntruMine* and other models, as well as the algorithm's time complexity analysis. Section 5 conducts the performance evaluations; Section 6 briefly comments on related works and Section 7 concludes the paper.

2 Problem Formulation

The CPS employs a large number of sensors to monitor areas. Such sensors are expected to be of low cost, small size and low transmission load. As a result, inexpensive passive sensors are often the best choice.

Though having different mechanisms and measurements, most passive sensors report the detected signals as a numeric value. For example, an acoustic sensor measures the air pressure of sound wave and a magnetic sensor generates the readings about magnetic force. Such measurements are influenced by two factors: (1) the intruder's energy (*i.e.*, the strength of emitted signals from the intruder); (2) the distance between the sensor and the intruder. Usually we can model the relationship between intruder o and sensor s as Eq. (2.1).

$$f(o,s) = \frac{e}{\alpha \cdot d(o,s)^{\beta} + \gamma} \tag{2.1}$$

where e is o's energy and $d(o,s)$ is the Euclidean distance between them. The parameters α, β and γ are determined by the sensor types and mechanisms.

If there are multiple intruders in the monitoring area, we assume that their signals aggregate at each sensor. Let O be the intruder set, sensor s's reading is estimated as Eq. (2.2).

$$\hat{r}(s) = \sum_{o \in O} f(o,s) = \sum_{o \in O} \frac{e}{\alpha \cdot d(o,s)^{\beta} + \gamma} \tag{2.2}$$

All real world signals are influenced by noise, hence the observed reading of s is

$$\begin{aligned} r(s) &= \hat{r}(s) + \epsilon = \sum_{o \in O} f(o,s) + \epsilon \\ &= \sum_{o \in O} \frac{e}{\alpha \cdot d(o,s)^{\beta} + \gamma} + \epsilon \end{aligned} \tag{2.3}$$

Without loss of generality, we assume that the background noise is zero mean Gaussian noise, *i.e.*, $\epsilon \sim N(0, \sigma^2)$.

Note that the sensor readings are collected and transmitted by various gateways, such as the aircraft bridges in Figure 1. There are many state-of-the-art works on gateway design, sensor deployment and message transmission [9, 3]. Since the main theme of this study is on data mining, we assume that the data can be collected by CPS (*i.e.*, the sensors have been already deployed and the command center can receive timeslice snapshots of the data in real time). Now the task boils down to find out the intruders' information from such data.

Problem Definition: Let S be the set of deployed sensors in a CPS, $S = \{s_1(x_{s1}, y_{s1}), s_2(x_{s2}, y_{s2}), s_3(x_{s3}, y_{s3}), \ldots, s_n(x_{sn}, y_{sn})\}$, and R be the sensors' readings in a snapshot, $R = \{r(s_1), r(s_2), \ldots, r(s_n)\}$. The task of *IntruMine* is to estimate the intruders $O = \{o_1(x_{o1}, y_{o1}, e_1), o_2(x_{o2}, y_{o2}, e_2), \ldots, o_k(x_{ok}, y_{ok}, e_k)\}$ based on S and R.

Notation	Explanation	Notation	Explanation
S	the sensor set	s, s_i, s_j	the sensors
O	the intruder set	o, o_i, o_j	the intruders
G	the monitoring graph set	g	the monitoring graphs
R	the reading set	$r(s)$	the reading of s
α, β, γ	the sensing parameters	$\tau(o)$	the confidence of o
δ_r	the reading threshold	δ_o	the confidence threshold
x, y	the 2D coordinates	e_i	the energy of o_i
θ	the attribute vector	Δ	the reading difference
ε	the background noise	σ	the standard deviation

Figure 2: List of Notations

We will propose intruder mining techniques in the next section. Figure 2 lists the notations used throughout this paper.

3 Intruder Mining

3.1 Monitoring Graph

Although the CPS has a large set of deployed sensors, not all of them are relevant to intruder mining. Typically, only a few of the sensors have detected signals from intruders. Thus the first step of *IntruMine* is to select the set of relevant sensors and make a rough initialization for the intruders. Let us examine an example in the system.

Example 1. Figure 3 shows a snapshot in CPS. The triangle nodes represent intruders and round ones are the deployed sensors. The sensors with relatively high readings are tagged with red color (*i.e.*, solid circle) and those with normal readings are in blue color (*i.e.*, shaded circle). Intruder o_1 is detected by sensors s_1, s_2, s_3, s_5 and s_6. s_1's reading is the highest since it is the closest to the intruder. s_4 should also detect o_1 but it reports a false negative reading. Intruder o_2 influences the readings of s_7, s_9, s_{10} and s_{11}. However, s_{12}, s_{13}, s_{14} and s_{15} are damaged by the harsh environment, they send out false positive readings.

Example 1 shows that the sensors near the intruders usually have higher readings than sensors that are far away: s_1 and s_7 both have higher readings than their neighbors, *i.e.*, their readings are local maxima. Such sensors are denoted as the *peak reading sensors*. They can be easily obtained by a single scan of the sensors' readings.

For each peak reading sensor s, *IntruMine* initial-

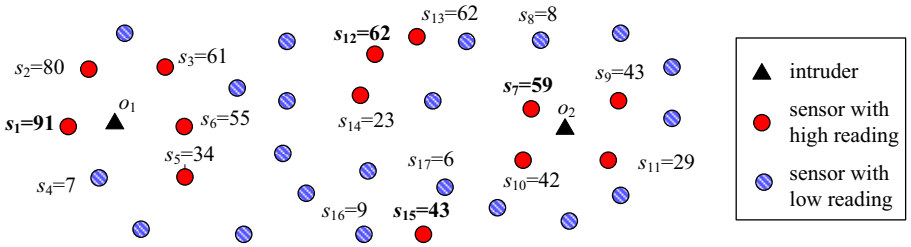

Figure 3: Example: A Snapshot in CPS

izes an intruder o at s's position and sets the energy of o as the peak reading. In some rare cases, the neighboring sensors may have the same readings, such as sensors s_{12} and s_{13} in Figure 3. Then the system randomly picks one of them to initialize the intruder. It is worth noting that other strategies can also be used to initialize the intruders, such as sampling method [16], random selection [14], and so on. However, the peak sensor-based method is the most effective way according to our experiments.

With the initialized intruders, the system can select relevant sensors to construct the relationship graph.

Definition 1. Let s be a sensor, o be an intruder and δ_r be the reading threshold. If $f(o, s) > \delta_r$, then s and o are in a monitoring-and-monitored relationship.

Intuitively, if s and o are in such a relationship, then s's reading is significantly influenced by o, *i.e.*, s captures a strong signal from o. Submitting Eq. (1) into the condition of Definition 1, we get the following corollary.

Corollary 1. Let o be an intruder, e be the intruder's energy and S be the sensor set, then the monitoring sensor set of o is

$$S_o = \{s | s \in S, d(o, s) < (\frac{e/\delta_r - \gamma}{\alpha})^{\frac{1}{\beta}}\}$$

Based on corollary 1, the system selects the monitoring sensors for each initialized intruder. If we use a link to represent the monitoring-and-monitored relationship, we can retrieve a set of *monitoring graphs*.

Figure 4 lists the detailed steps to construct the monitoring graphs. The algorithm first retrieves the peak sensor set S_p from the data (Lines 1 – 4). The intruders are then initialized by the information of peak sensor's position and reading (Lines 5 – 9). Finally, the system computes the monitoring sensors for each intruder and constructs monitoring graphs based on their relationships (Lines 10 – 15).

Example 2. Figure 5 shows the monitoring graphs retrieved from Example 1. Four intruders are initialized and their monitoring sensors are retrieved to construct

Algorithm 1. Monitoring Graph Construction
Input: sensor set S, reading set R at a snapshot, reading threshold δ_r.
Output: The monitoring graph set G.

```
1.  initialize peak sensor set S_p;
2.  for each sensor s of S
3.      if r(s) is larger than the readings of s's
        neighbors
4.          add s to S_p;
5.  initialize intruder set O;
6.  for each peak sensor s_p in S_p;
7.      initialize intruder o(x_o, y_o, e);
8.      x_o = x_sp, y_o = y_sp, e = r(s_p);
9.      add o to O;
10. for each intruder o of O
11.     retrieving the monitoring sensor set S_o;
        //Corollary 1
12.     initialize monitoring graph g;
13.     for monitoring sensor s_i of S_o
14.         add link l(o, s_i) to g;
15.     add g to G;
16. return G;
```

Figure 4: Algorithm: Constructing the Monitoring Graphs

the graph. There are more than 30 sensors deployed in the field, but the graphs filter out most of them and only 11 sensors are considered relevant in the constructed graphs.

The monitoring graph is a bipartite graph. The graph size is determined by the reading threshold δ_r. If δ_r is high, few sensors are selected and the graphs are tight. Otherwise, more monitoring sensors are included in the graph. Some graphs may even connect with each other. The monitoring graphs have two advantages: (1) They help select out a small set of relevant sensors that receive significant signals from the intruders; and (2) the graphs partition the reading set and separate most faulty readings from the real intruders. In Example 2, the real intruders o_1 and o_2 are in graphs g_1 and g_2, and most faulty readings are in g_3 and g_4. Hence

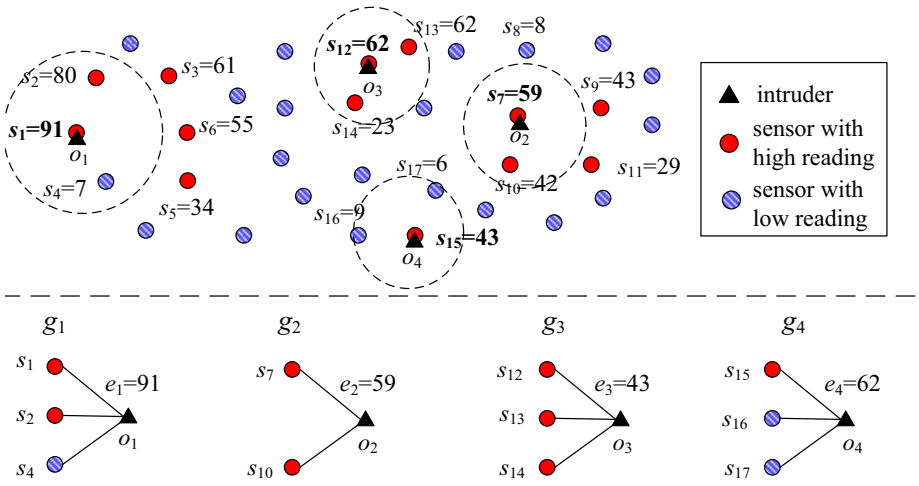

Figure 5: Example: The Monitoring Graphs

the influence of faulty readings on real intruders is significantly reduced.

3.2 Estimation of Attributes of Intruders

The monitoring graphs are constructed based on the initialization of the intruders. In other words, they are derived from the peak reading sensors. Hence the graphs may not be accurate. In Figure 5, g_1 misses sensors s_3, s_5 and s_6, g_2 should also include s_9 and s_{11}. To overcome this problem, we propose the method to adjust the attributes of intruders and refine the monitoring graphs iteratively.

In a monitoring graph g, the reading of each sensor, $r(s)$, is already known. With the information of intruders, we can further calculate the estimated reading of s as:

$$\hat{r}(s) = \sum_{o \in g} f(o, s) \tag{3.4}$$

Then the probability of observing the sensor's reading as $r(s)$ is

$$p(s) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r(s)-\hat{r}(s))^2}{2\sigma^2}} \tag{3.5}$$

The joint probability of observing the readings within monitoring graph g is

$$\begin{aligned} p(S) &= \prod_{s \in g} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r(s)-\hat{r}(s))^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{|g|} e^{-\frac{\sum_{s \in g}(r(s)-\hat{r}(s))^2}{2\sigma^2}} \end{aligned} \tag{3.6}$$

Hence the log-likelihood of observing the readings of sensors within monitoring graph g is

$$\begin{aligned} &\log(p(S)) \propto \\ &-|g| \log(\sigma^2) - \frac{\sum_{s \in g}(r(s) - \hat{r}(s))^2}{\sigma^2} \end{aligned} \tag{3.7}$$

Suppose there are k intruders in the monitoring graph g, the intruder's attribute vector $\theta = \{(x_{o1}, y_{o1}, e_1), (x_{o2}, y_{o2}, e_2), \ldots, (x_{ok}, y_{ok}, e_k)\}$. Based on the maximum likelihood criterion, the estimation of θ is equivalent to

$$\begin{aligned} &\arg\max_{\theta} - \sum_{s \in g} \left(r(s) - \hat{r}(s)\right)^2 \\ &= \arg\min_{\theta} \sum_{s \in g} \left(r(s) - \hat{r}(s)\right)^2 \end{aligned} \tag{3.8}$$

The difference between the estimated reading $\hat{r}(s)$ and the real reading $r(s)$ represents the error ϵ of intruder information. In the best case, the sensor's readings just fit intruder's information and the difference is minimized.

Let $\Delta_g = \sum_{s \in g}(r(s) - \hat{r}(s))^2$. We can use a gradient descent algorithm to compute the attribute vector θ iteratively. At the first iteration, θ is initialized with the information of peak reading sensors. The value of the n-th iteration can be calculated from the gradient of $(n-1)$-th iteration as follows.

$$\theta_i^n = \theta_i^{n-1} - \frac{1}{\sqrt{n}} \frac{\partial \Delta_g^{n-1}}{\partial \theta_i^{n-1}} \tag{3.9}$$

Let S_{oi} be the monitoring sensor set of intruder o_i, the gradient of Δ_g with respect to x_{oi}, y_{oi} and e_i are

Algorithm 2. Estimating Intruder's Attributes
Input: The initialized monitoring graph set G.
Output: The updated graph set G.

1. **for** each graph g of G
2. retrieve intruder set $O = \{o_1, o_2, \ldots, o_k\}$ from g;
3. $\theta^0 = \{(x_{o1}, y_{o1}, e_1), (x_{o2}, y_{o2}, e_2), \ldots, (x_{ok}, y_{ok}, e_k)\}$;
4. **repeat**
5. calculate the reading difference Δ_g;
6. calculate $\frac{\partial \Delta_g}{\partial x_{oi}}$, $\frac{\partial \Delta_g}{\partial y_{oi}}$, $\frac{\partial \Delta_g}{\partial e_i}$;
7. compute θ^n;
8. **for** each intruder o_i in O
9. update x_{oi}, y_{oi} and e_i according to θ^n;
10. retrieve new monitoring sensor set S_{oi}';
11. update graph g;
12. **until** stable;
13. **return** G;

Figure 6: Algorithm: Estimating the Intruder's Attributes

computed as shown in Eqs. (3.10) to (3.12).

$$\frac{\partial \Delta_g}{\partial x_{oi}} = \sum_{s_j \in S_{oi}} \frac{4\alpha(r(s_j) - \hat{r}(s_j)) \cdot e_i \cdot (x_{sj} - x_{oi})}{(\alpha d(o_i, s_j)^\beta + \gamma)^2} \tag{3.10}$$

$$\frac{\partial \Delta_g}{\partial y_{oi}} = \sum_{s_j \in S_{oi}} \frac{4\alpha(r(s_j) - \hat{r}(s_j)) \cdot e_i \cdot (y_{sj} - y_{oi})}{(\alpha d(o_i, s_j)^\beta + \gamma)^2} \tag{3.11}$$

$$\frac{\partial \Delta_g}{\partial e_i} = \sum_{s_j \in S_{oi}} \frac{-2(r(s_j) - \hat{r}(s_j))}{\alpha d(s_j, o_i)^\beta + \gamma} \tag{3.12}$$

Based on theoretical analysis, we design the algorithm to iteratively estimate intruder's attributes as shown in Figure 6. The algorithm first retrieves an intruder set and initializes the attribute vector θ based on original monitoring graphs (Lines 1 – 3). At each iteration, the system computes the reading differences and gradients to update θ (Lines 5 – 7). Meanwhile the algorithm regenerates new monitoring sensor set S'_{oi} and updates the graph g (Lines 8 – 11). After that, a new round of estimation is carried out on the updated graph. This process repeats until both the monitoring graph and the estimation are stable (Line 12).

Example 3. Figure 7 shows the running process of Algorithm 2. In the beginning, the intruders o_1, o_2, o_3 and o_4 are initialized with the information of peak reading sensors (*i.e.*, hollow triangles). The algorithm gradually adjusts the intruder's positions and energies during the process. New monitoring sensors are retrieved according to the updated intruders (*i.e.*, solid triangles). In the updated graph g_1, the sensors s_3, s_5 and s_6 are linked with intruder o_1. And the sensors s_9 and s_{11} are also included in g_2.

3.3 Verification of Trustable Sensors and Intruders

The estimation algorithm adjusts the intruder's attribute values and updates the monitoring graphs. However, there are still two problems influencing the mining accuracy: (1) some monitoring graphs are created due to faulty readings and may not contain any real intruders (*e.g.*, g_3 and g_4 in Figure 5). Hence the false positive intruders will be generated by such graphs. (2) Some sensors in the monitoring graph of real intruders are unreliable, such as s_4 in g_1. To solve this problem, the system needs to verify the estimated results.

From Eq. (3.7), the derivative of likelihood $\log(p(S))$ with respect to σ^2 is

$$\frac{\partial \log(p(S))}{\partial \sigma^2} = -\frac{|g|}{\sigma^2} - \frac{\sum_{s \in g}(r(s) - \hat{r}(s))^2}{(\sigma^2)^2} \tag{3.13}$$

Setting the derivative to zero, we obtain

$$\sigma^2 = \frac{\sum_{s \in g}(r(s) - \hat{r}(s))^2}{|g|} \tag{3.14}$$

Based on the estimated σ^2, we can verify the reliability of the sensor's reading. A classic measurement in statistics is the 3-standard deviation: If the deviation of estimated reading $\hat{r}(s)$ with respect to the true reading $r(s)$ is not within 3 standard deviations, *i.e.*, $(r(s) - \hat{r}(s))^2 > 9\sigma^2$, then we judge that the reading of such sensor is unreliable.

To filter out false positives, we define the confidence of intruder detection as follows.

Definition 2. The confidence of detected intruder o is the probability that o really exists, denoted as $\tau(o)$.

Intuitively, the readings of monitoring sensors are caused by intruders. If the actual readings are similar to the estimated ones, this suggests a high confidence that the intruder is real. For a false positive, the difference between actual and estimated readings will be large. Therefore, we can estimate the confidence of a detected intruder from the reading difference of its monitoring sensor set.

In the verification process, the system first calculates the reading difference Δ_{oi} for each intruder and

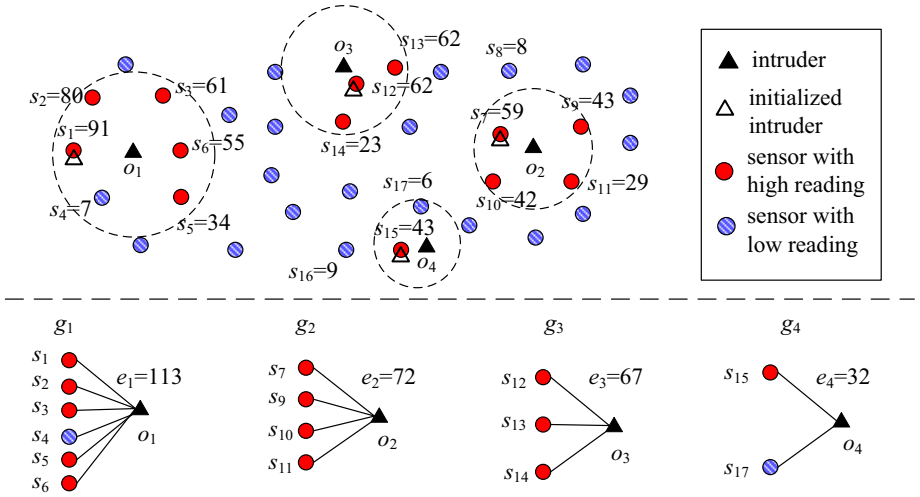

Figure 7: Example: The Estimation of Intruder's Attributes

the average difference of all the intruders.

$$\Delta_{oi} = \frac{\sum_{s \in S_{oi}} (r(s) - \hat{r}(s))^2}{|S_{oi}|} \tag{3.15}$$

$$\bar{\Delta} = \frac{\sum_{oi \in O} \Delta_{oi}}{|O|} \tag{3.16}$$

Then the intruder detection confidence is estimated as Eq. (3.17). For an intruder o_i, if the monitoring sensor's readings are coherent with the information (small reading difference), the intruder's confidence is high. If the reading difference is larger than $\bar{\Delta}$, which indicates that the real readings are quite different from the estimated ones. The intruder detection confidence is set as zero.

$$\tau(o_i) = \begin{cases} 1 - \frac{\Delta_{oi}}{\bar{\Delta}} (\Delta_{oi} < \bar{\Delta}) \\ 0 (\Delta_{oi} \geq \bar{\Delta}) \end{cases} \tag{3.17}$$

The entire process of intruder mining is summarized as Algorithm 3 in Figure 8. The system starts at generating monitoring graphs (Line 1). Then it estimates the intruder's attributes and meanwhile updates the structure of monitoring graphs (Line 2). In the verification step, the system filters out unreliable sensors since they may influence the computation of detected intruder confidence (Lines 3 – 5). Finally, it computes the confidence for each intruder and removes the unqualified ones (Lines 6 – 9).

Example 4. Figure 9 shows the example of intruder verification. After estimating the intruder's attributes, the system checks the sensors and removes s_4 from g_1. Then, the scores of intruder detection confidence are calculated. Suppose the confidence threshold δ_o is set as 0.5. The false positives o_3 and o_4 are filtered out, with o_1 and o_2 returned as the final result of intruder mining.

Algorithm 3. Intruder Mining
Input: sensor set S, reading set R at a snapshot, reading threshold δ_r, intruder confidence threshold δ_o
Output: the estimated intruder set O.

1. generate the monitoring graph set G w.r.t. R, S and δ_r; //Algorithm 1
2. estimate intruder's attributes and update G; //Algorithm 2
3. **for** each graph g of G
4. calculate the standard deviation σ;
5. filter out unreliable sensors;
6. retrieve intruders from G to O;
7. **for** each intruder o of O
8. compute $\tau(o)$;
9. **if** $\tau(o) < \delta_o$ **then** remove from O;
10. **return** O;

Figure 8: Algorithm: Mining the Intruders

4 Discussion

We presented the models and algorithms of *IntruMine*. Although it is a novel approach, *IntruMine* is well connected to some state-of-the-art models in literature. Here we briefly describe two existing techniques for intruder detection and show that both of them are special cases of *IntruMine*. We also study the time complexity of the proposed method and show the advantages of *IntruMine* over comparative methods based on complexity analysis.

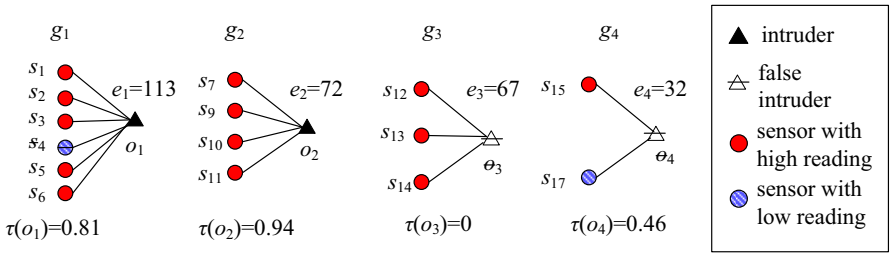

Figure 9: Example: Intruder Verification

4.1 Connection to Other Models

The principles of maximum likelihood (ML) based intruder detection method were proposed in [14]. This method formulates the task of intruder detection as a maximum likelihood estimation.

Let the reading set $R = (r(s_1), r(s_2), \ldots, r(s_n))$, the intruder energy $E = (e_1, e_2, , e_k)$, and the relation matrix

$$M = \begin{bmatrix} \frac{1}{\alpha d(o_1,s_1)^\beta+\gamma} & \frac{1}{\alpha d_(o_1,s_2)^\beta+\gamma} & \cdots & \frac{1}{\alpha d_(o_1,s_n)^\beta+\gamma} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{\alpha d_(o_k,s_1)^\beta+\gamma} & \frac{1}{\alpha d_(o_k,s_2)^\beta+\gamma} & \cdots & \frac{1}{\alpha d_(o_k,s_n)^\beta+\gamma} \end{bmatrix}$$

The intruder attribute vector θ is denoted as $\{x_{o1}, y_{o1}, e_1, x_{o2}, y_{o2}, e_2, \ldots, x_{ok}, y_{ok}, e_k\}$, the problem is then formulated to estimate θ that

$$\begin{aligned} &\arg\max_\theta - ||R - EM||^2 = \\ &\arg\min_\theta ||R - EM||^2 \end{aligned} \tag{4.18}$$

Eq.(4.18) is actually a special case of *IntruMine* by setting the reading threshold δ_r to 0. In the monitoring graph construction step, when the system retrieves monitoring sensor set S_o of intruder o, for each sensor $s \in S$,

$$f(o,s) = \frac{e}{\alpha \cdot d(o,s)^\beta + \gamma} > \delta_r = 0 \tag{4.19}$$

Then a link of s and o should be constructed in the monitoring graph g. In this way, eventually g includes all the sensors of S and Eq. (3.8) in Section 3.2 turns out to be:

$$\begin{aligned} &\arg\max_\theta - \sum_{s\in g} (r(s) - \hat{r}(s))^2 \\ &= \arg\min_\theta \sum_{s\in g} (r(s) - \hat{r}(s))^2 \\ &= \arg\min_\theta \sum_{s\in S} (r(s) - r'(s))^2 \\ &= \arg\min_\theta ||R - EM||^2 \end{aligned} \tag{4.20}$$

Hence *IntruMine* is reduced to the ML model. In such case, there is only one monitoring graph g constructed. This graph contains all the sensors and the initialized intruders, with links between each pairs of a sensor and an intruder. In the step of estimating the intruder's attributes, no matter how the position and energy of intruder o changes, for each sensor s linked with o, $f(o,s) > \delta_r = 0$. Then no link of g will be removed and the structure of g stays unchanged during the estimation process. Algorithm 2 is thus degenerated to the maximum likelihood estimation on a fixed graph.

The TruAlarm method is introduced in [16]. This method samples out several positions to initialize the intruders. TruAlarm links the intruders with alarming sensors within a fixed distance threshold δ_d and constructs the object-alarm graphs. The alarm trustworthiness analysis is carried out on such graphs.

The object-alarm graphs of TruAlarm can be deduced from *IntruMine*'s monitoring graph by setting certain constraints and premises: (1) The intruder's locations are constrained to a set of sampling points; and (2) all the intruders have the same energy level. When computing the monitoring sensor set for each intruder, the system can select the sensors within a fixed distance threshold $\delta_d = (\frac{e/\delta_r - \gamma}{\alpha})^{\frac{1}{\beta}}$.

Note that, both ML and TruAlarm methods detect the intruders with some premises: ML requires the number of intruders as input, and TruAlarm needs the distance threshold and possible intruder locations for sampling. However in real applications, it is hard for users to provide such parameters in advance. Incorrect inputs may also bring down the detection accuracy. It is a significant step forward for *IntruMine* to detect intruders without those premises.

4.2 Complexity Analysis

Generally speaking, *IntruMine* has three steps to process the data: (1) Constructing the monitoring graph and initializing the intruders (Algorithm 1); (2) Estimating the intruder attributes (Algorithm 2); and (3) Verifying the estimated results (Lines 3 – 10 in

Algorithm 3). We analyze the time complexity of each step as follows.

Proposition 1: Let N be the number of sensors, k be the number of estimated intruders, n be the total number of monitoring sensors, and l be the number of iterations in attribute estimation step. The time complexity of step 1 is $O(NlogN)$; step 2 is $O(k(n + logN)l)$, and step 3 is $O(kn)$.

Proof: In step 1, the system has to scan all the sensors and check each one's neighbors to find out the peak sensors. This process takes $O(NlogN)$ with a spatial index such as R-tree (since all the sensors have been deployed and their positions are known in advance, it is easy to build a spatial index). The system then initializes k intruders from the peak reading sensors and generates their monitoring sensor set with $O(klogN)$ time. The total time cost is $O(NlogN + klogN)$. Since k is much smaller than N, hence the time complexity of step 1 is $O(NlogN)$.

In step 2, the algorithm runs l iterations to reach a stable state. At each iteration, the algorithm has to compute the difference between real reading and estimated reading of each monitoring sensor. In the worst case, it has to scan k intruders to estimate a sensor's reading. There are n monitoring sensors in total. The time cost of attribute estimation is $O(kn)$. After estimating the attributes, the algorithm also needs to update the monitoring graphs with $O(klogN)$ time. Hence the total time complexity of step 2 is $O(k(n + logN)l)$.

In step 3, the algorithm computes the standard deviation of every sensor and the confidence score for each intruder. Both processes require to calculate the differences between real reading and estimated reading of the monitoring sensors, which takes $O(kn)$ time. Then the time complexity of step 3 is $O(kn)$. ∎

From Proposition 1, one can clearly see that the main time cost is at step 2. Since N is fixed and k is usually a small number. The key to improve algorithm's efficiency is to generate tight monitoring graphs, *i.e.*, reduce the number of monitoring sensors, n. With a reasonable reading threshold δ_r, *IntruMine* only selects the monitoring sensors that are significantly influenced by each intruder. In the ML method, δ_r is set to 0 and the system takes count in all the sensors, *i.e.*, $n = N$. The time complexity for maximum likelihood estimation is degenerated to $O(kNl)$. In addition, the majority of such sensors do not receive strong signals from the intruders and cannot contribute to the estimation. The system needs to take more iterations to reach a stable state (with larger l). Hence the efficiency of ML is much worse than *IntruMine*.

TruAlarm constrains the intruders to a sampling set of fixed locations. If the algorithm wants to achieve a high accuracy, it has to generate a large location set and initializes the intruder at every sampling location. In this way, the number of intruders, k, is much larger. Since the algorithm needs to generate the monitoring sensor set for each intruder, more sensors are inevitably involved in the constructed graphs. The algorithm efficiency is thus brought down due to larger monitoring graphs. However, the detection accuracy of TruAlarm may still be a problem. In [16], the authors report that more that 80% of the sampled intruders does not exist in the end. The peak reading sensor-based initialization of *IntruMine* is more efficient and effective.

5 Performance Evaluation

5.1 Experiment Setup

Datasets: We conduct extensive experiments to evaluate the proposed methods, using both real-world and synthetic datasets. To test the performance of *IntruMine* in large and untrustworthy data collected from CPS, we generate three synthetic datasets based on the military trajectories in the CBMANET project [6], in which an infantry battalion of 64 vehicles moves from Fort Dix to Lakehurst during a mission lasting 3 hours. The data generator simulates monitoring fields along their routes with 400 to 10,000 deployed sensors, and each sensor reports a reading every 10 seconds.

Baselines: The proposed *IntruMine* algorithm (IM) is compared with TruAlarm method (TA) in [16]. We also implement the maximum likelihood based estimation (ML) method based on the principles proposed in [14]. We evaluate both efficiency and effectiveness of the algorithms in the experiments.

Environments: The experiments are conducted on a PC with Intel 7500 Dual CPU 2.20G Hz and 3.00 GB RAM. The operating system is Windows 7 Enterprise. All the algorithms are implemented in Java on Eclipse 3.3.1 platform with JDK 1.5.0. The datasets and parameter settings are listed in in Figure 10.

Dataset	Object	Sensor. #	Reading. #	Faulty%
Real (D_1)	5	213	$2.1*10^5$	~10%
Syn 1 (D_2)	64	400	$4.3*10^5$	20%
Syn 2 (D_3)	64	2,500	$2.7*10^6$	30%
Syn 3 (D_4)	64	10,000	$1.1*10^7$	40%
δ_r: 0.3 – 0.9, default 0.7;		δ_o: 0.2 – 0.8, default 0.6; α=10, β=1, γ=1;		

Figure 10: Experiment Settings

5.2 Comparisons in Mining Efficiency

In the first experiment, we evaluate the efficiency of different algorithms on default settings. The system processes IM, TA and ML on the four datasets and records their average time cost on each snapshot. Figure 11(a) shows the results by dataset and Figure 11(b) records the algorithm's running time *w.r.t.* the total number of sensors, N. Note that the y-axes are in logarithmic scale. IM achieves the best efficiency on all the datasets. In the largest dataset D_4, IM is an order of magnitude faster than ML and two orders of magnitude faster than TA.

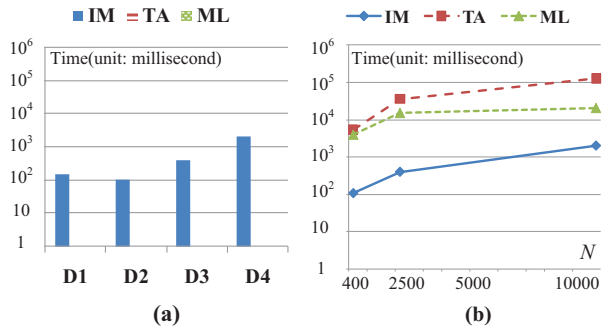

Figure 11: Efficiency: (a) different datasets and (b) sensor number

We also study the factors that influence IM's efficiency. We set the reading threshold δ_r as 30% to 90% of the typical intruder energy and record the algorithm's time cost on datasets D_1 to D_4 in Figure 12(a). Then we carry out the same experiments for confidence threshold δ_o (Figure 12(b)). The results show that the influence of δ_r is larger than δ_o, because δ_o only influences the verification step, but δ_r determines the size the monitoring graphs. With higher δ_r, the system can generate smaller monitoring graphs and increase the mining efficiency, especially in the large datasets with densely deployed sensors (*e.g.*, D_4).

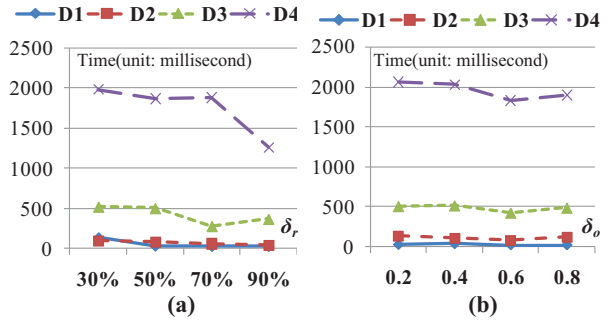

Figure 12: Efficiency: *IntruMine* on different δ_r and δ_o

5.3 Evaluations of Detecting Effectiveness

To evaluate the quality of mining results, we retrieve the intruder's true position and energy in each snapshot as the ground truth and compare the mining results with them. The system first compares the number of detected intruders with ground truth to calculate the measures of precision and recall, then matches the detected intruder to the nearest one in ground truth and computes the relative errors of energy and position.

Since ML requires the exact number of real intruders as the input (*i.e.*, with 100% precision and recall). We only record the precision and recall of IM and TA in Figure 13. Both of them can detect all the real intruders, but IM's precision is about 20% higher. The number of false positives reported by IM is only as half as TA, because IM's peak sensor based initialization and intruder verification step could filter out the false positives effectively.

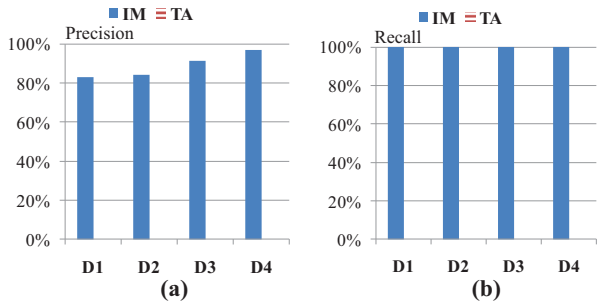

Figure 13: Effectiveness: (a) precision and (b) recall on different datasets

All three methods can detect the intruders, we further check their detecting effectiveness by calculating the relative errors of energy and position. The results are shown in Figure 14. ML has the largest errors: The average energy error is more than 50% and the position error is about 40% in D_3 and D_4. The reason of ML's failure is that the algorithm takes count in the reading from all the sensors, and it is inevitably influenced by the faulty readings and noises. ML's accuracy degenerates rapidly on dataset D_3 and D_4, since there are more faulty readings. This result indicates that ML is not feasible to process untrustworthy data. The errors of IM are much lower, with no more than 5% in all the datasets. The performance of IM even improves on D_3 and D_4, because with more deployed sensors, IM can effectively filter the untrustworthy ones and utilize the information for accurate estimation.

Then we investigate IM's effectiveness with different reading thresholds δ_r. Figure 15 shows IM's precision and recall on the four datasets *w.r.t.* different δ_r. When increasing δ_r from 30% to 50%, IM has a rapid increase

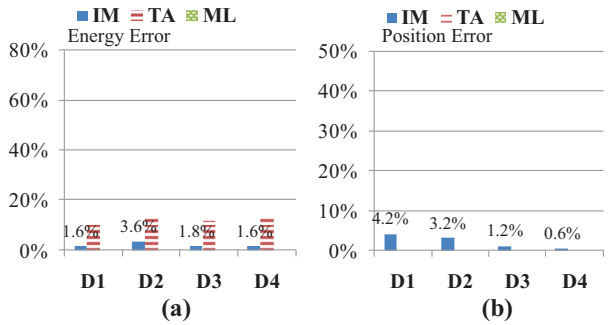

Figure 14: Effectiveness: (a) energy error and (b) position error on different datasets

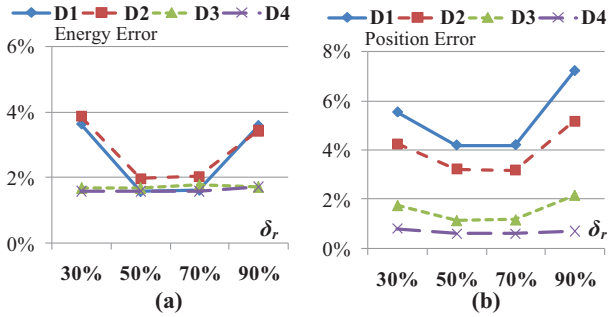

Figure 16: Effectiveness: (a) energy error and (b) position error *w.r.t.* δ_r

in precision, but the improvement is not significant with even larger δ_r. Figure 16 records the energy and position errors in the experiments. It is interesting to see that both errors are high with extreme δ_r. The reason is that, if δ_r is too low, the monitoring graphs may involve many irrelevant sensors that affect the attribute estimation. However, when δ_r is too high, there are very few monitoring sensors in the graph, the information might not be enough for accurate estimation. Hence, δ_r should be set in a moderate range (from 0.5 to 0.7) to achieve the best performance.

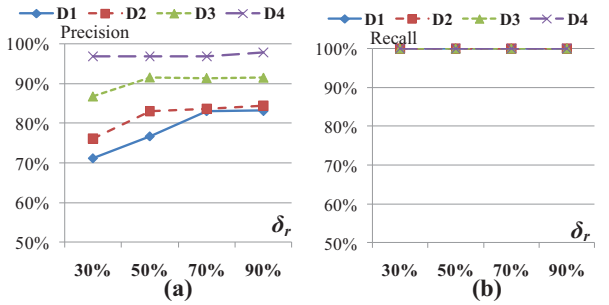

Figure 15: Effectiveness: (a) precision and (b) recall *w.r.t.* δ_r

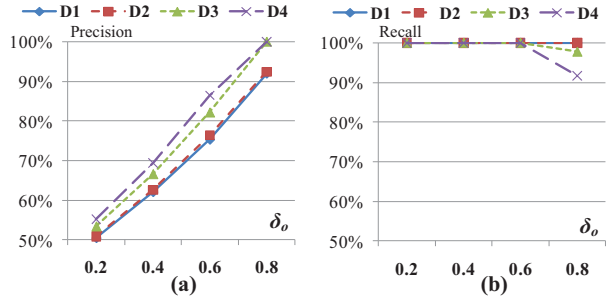

Figure 17: Effectiveness: (a) precision and (b) recall *w.r.t.* δ_o

Finally, we tune the confidence threshold δ_o and show the results in Figure 17 and 18. The major function of δ_o is to filter false positives. As δ_o grows, the precision increases. However, if δ_o is too high, some real intruders will also be filtered out and IM cannot guarantee 100% recall. The energy and position errors also increase rapidly with extreme δ_o, since there are not enough detected intruders to be matched with real ones. As a result, δ_o should not be set more than 0.8.

6 Related Work

Arora *et al.* propose the intrusion detection problem in wireless network and design a detection model with acoustic and magnetic sensors [1]. Lin *et al.* propose a framework for the in-network intruder tracking [9]. Li *et al.* propose the techniques of detecting blackhole and volcano patterns in directed networks, which is a significant improvement in this area [8, 7]. Cevher and Kaplan *et al.* study the problem of how to assign sensors to different tracking and monitoring tasks to achieve the optimal efficiency [3]. They propose a sensor assignment algorithm with fuzzy location estimation.

The main concerns of the above methods are about the sensor's energy and communication bandwidth. The solutions focus on providing an optimal sensor deployment plan. The task of *IntruMine* is different. But *IntruMine* actually complements those technologies and improves the system's applicability.

Hammad *et al.* propose the stream window join algorithm to track moving objects in sensor network database [5]. Aslam *et al.* propose a geometry-based method to track the intruders [2]. Ozdemir *et al.* use the techniques of particle filtering to track intruders [11]. Sheng and Hu propose the maximum likelihood-based estimation method [14] and Tang *et al.* propose the TruAlarm filtering method [16]. In most methods, the user has to provide some key parameters in advance, such as the intruder's moving speed, energy or the

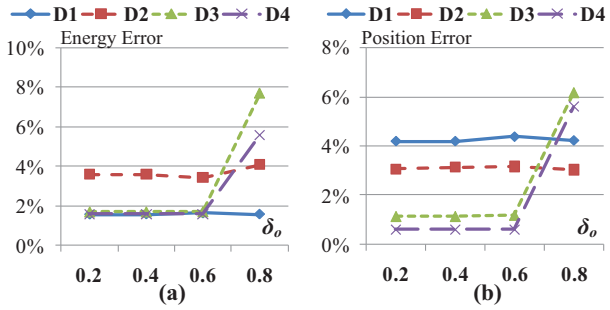

Figure 18: Effectiveness: (a) energy error and (b) position error *w.r.t.* δ_o

number of intruders. Many studies also make the assumption that there is no unreliable sensor data.

Pan *et al.* use the supervised learning method to locate Receiving Signal Sensors (RSS)[13]. The transfer learning techniques are proposed as semi-supervised mining [12]. The main difference between those works and *IntruMine* is about the sensors. The RSS are moving sensors that receive signals from some fix nodes (APs). And the algorithm is designed to learn the sensor's locations. In the CPS applications, most sensors are fixed and their locations are already known. The problem is to detect the location of intruders. In addition, the techniques in [13, 12] are supervised/semi-supervised learning. However, *IntruMine* is an unsupervised method.

7 Conclusion and Future Work

This paper studies the problem of intruder mining in untrustworthy data of cyber-physical systems. A method called *IntruMine* is proposed to detect and verify the intruders. In the *IntruMine* framework, the system constructs the monitoring graphs and estimates the intruder attributes with the link information of such graphs. The information of reading difference is used to filter out the unreliable sensors and false positives. We prove that two existing models are special cases of *IntruMine* with certain constraints. Extensive experiments are conducted, which shows the scalability and applicability of the proposed method.

There are many promising directions that can be explored further, such as predicting intruder's movement speed and directions beyond simple existence detection. We also plan to extend *IntruMine* to other application domains including environmental monitoring and traffic control.

8 Acknowledgements

The work was supported in part by U.S. NSF grants IIS-0905215, CNS-0931975, CCF-0905014, IIS-1017362, the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] A. Arora, P. Dutta, and S. Bapat. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[2] J. Aslam, Z. Butler, F. Constantin, and V. Crespi. Tracking a moving object with a binary sensor network. In *SenSys*, 2003.

[3] V. Cevher and L. M. Kaplan. Acoustic sensor network design for position estimation. In *ACM Transactions on Sensor Networks*, 2007.

[4] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities and challenges. In *Proceedings of The IEEE,*, 2003.

[5] M. A. Hammad, W. G. Aref, and A. K. Elmagarmid. Stream window join: Tracking moving objects in sensor network databases. In *SSDBM*, 2003.

[6] T. Krout. Cb manet scenario data distribution. In *BBN Tech. Report*, 2007.

[7] Z. Li, H. Xiong, and Y. Liu. Mining blackhole and volcano patterns in directed graphs: A general approach. In *Data Mining and Knowledge Discovery Journal*, 2012.

[8] Z. Li, H. Xiong, Y. Liu, and A. Zhou. Detecting blackholes and volcanoes in directed networks. In *ICDM*, 2010.

[9] C. Lin, W. Peng, and Y. Tseng. Efficient in-network moving object tracking in wireless sensor network. *IEEE Transaction on Mobile Computing*, 5(8), 2006.

[10] C. Lo and W. Peng. Carweb: A traffic data collection platform. In *MDM*, 2008.

[11] O. Ozdemir, R. Niu, and P. K. Varshney. Tracking in wireless sensor network using particle filtering: Physical layer considerations. In *IEEE Trans. on Signal Processing*, 2009.

[12] R. Pan, J. Zhao, V. W. Zheng, and J. J. Pan. Domain constrained semi-supervised mining of tracking models in sensor networks. In *KDD*, 2007.

[13] S. J. Pan, J. T. Kwok, Q. Yang, and J. J. Pan. Adaptive localization in a dynamic wifi environment through multi-view learning. In *AAAI*, 2007.

[14] X. Sheng and Y. Hu. Maximum likelihood multiple source localization using acoustic energy measurements with wireless sensor networks. In *IEEE Trans. on Signal Processing*, 2005.

[15] R. Szewczyk, J. Polastre, and J. Mainwaring. Lessons from a sensor network expedition. In *European Workshop on Wireless Sensor Networks*, 2004.

[16] L. Tang, X. Yu, S. Kim, J. Han, C. Hung, and W. Peng. Tru-alarm: Trustworthiness analysis of sensor networks in cyber-physical systems. In *ICDM*, 2010.

[17] G. Tolle, J. Polastre, and R.Szewczyk. A macroscope in the redwoods. In *Sensys*, 2005.

[18] R. Tomkins. Thermo eye helps keep peace. In *Word Press*, 2007.

[19] Y. Zheng and X. Zhou. *Computing with Spatial Trajectories*. Springer, 2011.