

# Tru-Alarm: Trustworthiness Analysis of Sensor Networks in Cyber-Physical Systems

Lu-An Tang<sup>1</sup>, Xiao Yu<sup>1</sup>, Sangkyum Kim<sup>1</sup>, Jiawei Han<sup>1</sup>, Chih-Chieh Hung<sup>2</sup>, and Wen-Chih Peng<sup>2</sup>

<sup>1</sup>Department of Computer Science, University of Illinois at Urbana-Champaign

<sup>2</sup>Department of Computer Science, National Chiao Tung University

{tang18, xiaoyu1, kim71}@illinois.edu, hanj@cs.uiuc.edu, {wcpeng, hungcc}@cs.nctu.edu.tw

**Abstract**—A *Cyber-Physical System (CPS)* integrates physical devices (e.g., sensors, cameras) with cyber (or informational) components to form a situation-integrated analytical system that responds intelligently to dynamic changes of the real-world scenarios. One key issue in CPS research is trustworthiness analysis of the observed data: Due to technology limitations and environmental influences, the CPS data are inherently noisy that may trigger many false alarms. It is highly desirable to sift meaningful information from a large volume of noisy data. In this paper, we propose a method called *Tru-Alarm* which finds out trustworthy alarms and increases the feasibility of CPS. *Tru-Alarm* estimates the locations of objects causing alarms, constructs an *object-alarm graph* and carries out trustworthiness inferences based on linked information in the graph. Extensive experiments show that *Tru-Alarm* filters out noises and false information efficiently and guarantees not missing any meaningful alarms.

## I. INTRODUCTION

The research of *Cyber-Physical System (CPS)* has received increasing attention in recent years. This topic is placed with top priority in the federal research investment list of U.S. President's council of advisors on science and technology [9]. The CPSs collect sensor data from physical world and link them to various information sources for real-time analysis. Such systems are widely applied in the areas of highway traffic monitoring [7], healthcare systems [10], [11] and battlefield surveillance [1].

A reliable CPS should reach a high trust level. However, the data trustworthiness issue is listed as a major challenge for CPS applications, partly due to the following difficulties:

- **Huge Data Size:** A typical CPS includes hundreds of sensors and each sensor generates data records in every few minutes. The management system is required to process the huge data set and evaluate the alarms with high efficiency.
- **Non-Availability of Training Sets:** Many traditional false alarm detection methods are supervised, *i.e.*, their classifiers are built based on training datasets [8], [6], [13]. Such aids are hard to get in CPS since it is costly and error-prone to manually label the large sensor dataset.
- **Conflicts of Sensors:** A well deployed CPS system has reasonable redundancies. In case that a sensor does not work correctly, the others still provide accurate information. Thus conflicts frequently occur among the reliable and faulty sensors. The challenge is that the system does

not know which sensor is trustworthy in advance. The truth needs to be inferred from conflicting data.

**Example 1.** A recent article [1] reports that U.S. forces are using sensor network systems in Iraq to protect troops and bring security to towns. The so-called *battle-network* systems constantly watch designated areas day and night, detect approaching enemies and send alarms to command center. Figure 1 shows a deployment of such system. It has 28 sensors around a troop station. Suppose the alarm threshold is set to 10. If a sensor's reading is larger than the threshold, an alarm is generated. In this example, an enemy soldier enters the monitored region and he is detected by surrounding sensors  $s_4, s_5, s_6, s_7$  and  $s_8$ . Sensor  $s_7$ 's reading is the highest since it is closest to the soldier. The five surrounding sensors send alarms to command center. However, the deployed sensors are easily damaged in harsh environments. Some irrelevant activities, such as windblown debris or animal movements, also influence the sensors and may cause false alarms. In this example, a mouse passes in front of sensor  $s_{15}$ , its activity causes a high reading of the sensor and generates a false alarm.

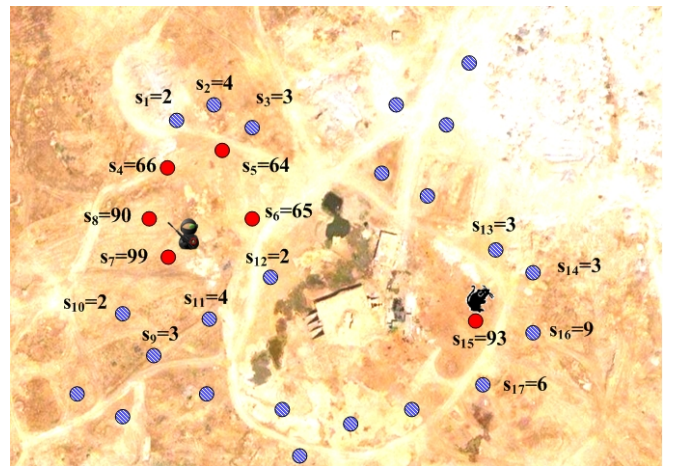


Figure 1. Battle-Network System Example

Many traditional methods detect false alarms based on the *neighborhood similarity hypothesis* [3], [8], [5]. This hypothesis assumes that a sensor's reading should be similar to its  $k$ -Nearest Neighbors ( $k$ NNs). The alarm is likely to be false if its reading is different from the  $k$ NNs. However,

reliable sensors also cannot detect the object's activity if it is distant. The false negatives (missing meaningful alarms) may be caused by this hypothesis. In Example 1,  $s_5$ 's four nearest neighbors are  $s_3, s_2, s_1$  and  $s_4$ , but only  $s_4$ 's reading is close to  $s_5$ , the other three's readings are much lower than  $s_5$ . According to the neighborhood similarity hypothesis,  $s_5$ 's alarm is dismissed.

In this study, the authors propose a novel approach, *Tru-Alarm*, to infer meaningful and trustworthy alarms. It is unsupervised yet effective. Different from traditional sensor oriented approaches, *Tru-Alarm* takes the objects that cause alarm into consideration. The system builds up an object-alarm graph to model the relationship between alarms and intrusion objects. The trustworthiness analysis is carried out on such graph. Our contributions can be summarized as follows.

- Introducing the concepts of trustworthiness scores to measure the object and alarm confidence in CPS.
- Proposing a framework to find the trustworthy alarms: The system first estimates locations of intrusion objects, then constructs the *object-alarm graph* to connect the alarms with related objects. The trustworthiness analysis tasks are explored along the links of such graph.
- Designing pruning techniques to help efficient trustworthiness inference. The search space of object-alarm graph is large. The object pruning algorithm is designed to bring down the large graph size.
- Conducting extensive experiments to evaluate the effectiveness and efficiency of proposed methods. The results show that *Tru-Alarm* yields higher precision and recall than existing methods.

The rest of the paper is organized as follows. Section 2 introduces the preliminary knowledge and defines the problem; Section 3 describes the framework of *Tru-Alarm*; Section 4 presents the techniques of efficient trustworthiness inference; Section 5 conducts the performance evaluations; and Section 6 concludes the paper.

## II. BACKGROUND AND PRELIMINARIES

The CPSs are deployed in different scenarios with various types of sensors. For example, infrared sensors, ultrasonic sensors and acoustic sensors are frequently used to detect intrusion objects. They have different sensing mechanisms and measurements. In this study, we use a general term, *severity*  $r(s, t)$ , to represent the detected signal strength by sensor  $s$  at time  $t$ . Assuming there is no obstacle between sensor  $s$  and object  $o$ , the equation of  $r(s, t)$  can be written as:  $r(s, t) = f(\text{dist}(s, o), \Omega(o))$ , where  $\Omega(o)$  is object  $o$ 's signal strength.

Given a severity threshold  $\delta_s$ , if a data record's severity  $r(s, t) > \delta_s$ , an alarm is generated and this record is tagged as alarming record, denoted as  $r_a(s, t)$ . In CPS, the alarming record set  $R_a$  is a subset of the entire dataset  $R$ .

Localizing object's positions from sensor's readings has been a hot research topic in the past decade. Many state-of-the-art methods have been proposed in [2], [12], [4]. A common applied work is the sampling approach. Suppose the possible regions of alarming sensors cover  $N$  positions, with a sample ratio  $l\%$ , the sampling algorithm uniformly selects  $n = l\% \times N$  points as the possible object locations. **Example 2.** Figure 2 shows the battle-network deployment of Example 1. There are six sensors reporting alarms. The possible regions are bounded by circles of dashed lines. Suppose those regions cover 105 points and the sampling ratio is 10%. The algorithm selects 10 possible objects uniformly distributed in those regions. Such sampling method may generate many false positives (the objects that do not really exist). The work is left to *Tru-Alarm* to find out trustworthy objects and filter the non-existing ones.

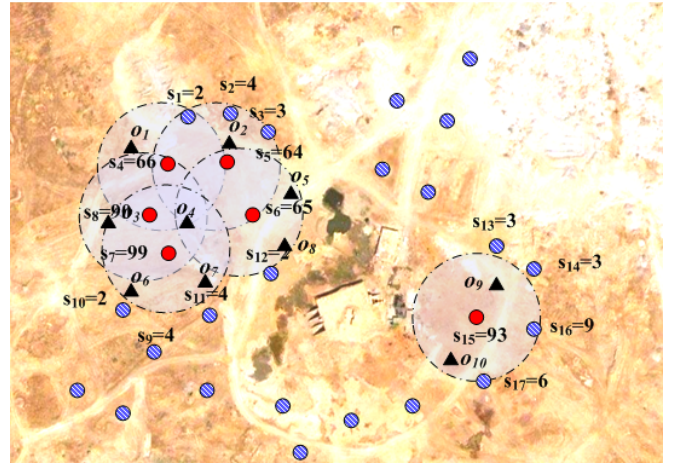


Figure 2. Sampled Objects on Battle-network System

To quantitatively measure the trustworthiness in CPS, we introduce some basic concepts as follows.

**Definition 1.** Let  $r_a(s, t)$  be an alarming record generated by sensor  $s$  at time  $t$ , the trustworthiness of  $r_a(s, t)$  is the probability of it being correct, denoted as  $\tau(r_a(s, t))$ .

**Definition 2.** The trustworthiness of an object  $o$  is the probability that  $o$  really exists, denoted as  $\tau(o)$ .

Note that, a trustworthy object is the one that is meaningful to users. In Example 1, sensor  $s_{15}$  actually detects the object as a mouse. However, the objects in battle-network should be intrusion enemies, a mouse is not interesting to users.

The task of *Tru-Alarm* is to find out the trustworthy alarms and meaningful objects, which can be formally described as follows.

**Problem Definition.** Let  $R = \{r(s_1, t_1), r(s_1, t_2), \dots, r(s_m, t_n)\}$  be a CPS dataset,  $R_a \subseteq R$  be the set of alarm records and  $O$  be the object set generated by localization method, given a trustworthy threshold  $\delta_t$ , the *Tru-Alarm*'s task is to find out the trustworthy alarms with  $\tau(r_a(s, t)) > \delta_t$  and meaningful objects with  $\tau(o) > \delta_t$ .

### III. THE FRAMEWORK OF TRU-ALARM

It is easy for a user to learn the sensor's detecting ranges before CPS deployment. If a sensor works well, it should detect any object movements inside the range. Formally, we define the monitored object set of a sensor as below.

**Definition 3.** Let  $O$  be the object set and  $d_s$  be the detecting range of a sensor  $s$ . The monitored object set of  $s$  is defined as  $O_s = \{o \mid o \in O, \text{dist}(s, o) < d_s\}$ .

Similarly, we obtain the monitoring sensor set.

**Definition 4.** Let  $S$  be a sensor set and  $d_s$  be the detecting range of a sensor  $s$ . The monitoring sensor set of an object  $o$  is defined as  $S_o = \{s \mid s \in S, \text{dist}(s, o) < d_s\}$ . If  $s \in S_o$  generates an alarm record  $r_a(s, t)$ , then  $r_a(s, t)$  is said to be related to  $o$ .

One may notice that, an alarm may be related to multiple objects, and an object usually has several related alarms. In this way, we build up a relational graph between the objects and alarms. In this *object-alarm* graph, two kinds of nodes are presented: the objects and their monitoring sensor's data records. The monitoring-monitored relationship is modeled as an edge in the graph.

**Example 3.** Figure 3 (a) shows the monitoring sensor set retrieved from Example 2. The red nodes are the alarming sensors, the blue nodes represent normal sensors and the triangle nodes are the objects. For instance, object  $o_2$  has three monitoring sensors:  $s_2$ ,  $s_3$  and  $s_5$ , in which only  $s_5$  reports an alarm  $r_a(s_5, t)$ . It is related to  $o_2$ . Meanwhile  $r_a(s_5, t)$  is also related to  $o_4$  and  $o_5$ , since the three objects are all in  $s_5$ 's detecting range and any of them may cause the alarm. Figure 3 (b) shows the corresponding object-alarm graph.

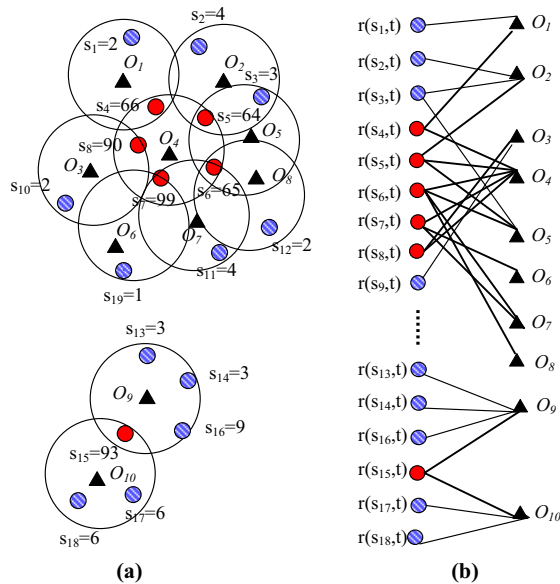


Figure 3. Monitoring Sensors and Object-Alarm Graph

There are two kinds of edges in the graph: the *normal*

*edge* linking the object with a normal record, and the *object-alarm edge*. The *weight* of an object-alarm edge represents the probability that the alarm is caused by such object. Hence the first step of trustworthiness inference is to infer the weights of object-alarm edges, *i.e.*, compute the conditional alarm trustworthiness  $\tau(r_a(s_i, t)|o)$ .

$\tau(r_a(s_i, t)|o)$  is determined by the coherence of other sensors' readings in monitoring sensor set  $S_o$ . If other sensor's readings are all coherent with  $r_a(s, t)$ , its trustworthiness is high, otherwise it is unlikely to be caused by  $o$ .

$$\tau(r_a(s_i, t)|o) = \frac{\sum_{s_j \in S_o, s_j \neq s_i} \text{coh}(r(s_j, t), r_a(s_i, t))}{|S_o| - 1} \quad (1)$$

To estimate the coherence score between two sensors' records, the system should take count in both their severity difference and positions. For example,  $s_5$  and  $s_7$  in Figure 3 have a large severity difference, but they are actually coherent if counting in the distance factor. When computing  $\text{coh}(r_a(s_i, t), r(s_j, t))$ , the system should consider whether  $s_j$  would report the same severity if it was located at  $s_i$ 's position.

As mentioned in Section 2, the equation of computing  $r(s, t)$  can be learned in advance. Then it is easy to deduce the inverse function of object  $o$ 's signal strength. The expected severity of sensor  $s_j$  at  $s_i$ 's location is computed as

$$r'(s_j, t) = f(\text{dist}(s_i, o), \Omega_j(o))$$

Coherence  $\text{coh}(r_a(s_i, t), r(s_j, t))$  is judged by the difference of the expected severity and real reading of  $s_j$ , as shown in Eq. (2). Its value range is  $[0, 1]$ . A standard deviation  $\sigma$  is computed for all the sensors in monitoring sensor set  $S_o$ . If sensor  $s_j$ 's severity is the same as expected value, the coherence score reaches the maximum of 1; if the difference is larger than standard deviation  $\sigma$ , *i.e.*,  $s_j$ 's severity is quite different from expected value, the coherence score is set to 0.

$$\text{diff}(r', r) = |r'(s_j, t) - r(s_j, t)|$$

$$\text{coh}(r_a(s_i, t), r(s_j, t)) = \begin{cases} 1 - \frac{\text{diff}(r', r)}{\sigma} & \text{if } \text{diff}(r', r) < \sigma \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

A low  $\tau(r_a(s, t)|o)$  indicates two possibilities: (i)  $r_a(s, t)$  is a false alarm; or (ii)  $r_a(s, t)$  is a true alarm, but it is not caused by object  $o$ . In either case, object  $o$ 's trustworthiness should be decreased. Therefore, we compute object  $o$ 's trustworthiness as the average of all its conditional alarm trustworthiness in Eq. (3).

$$\tau(o) = \frac{\sum_{r_a \in R_a} \tau(r_a(s, t)|o)}{|S_o|} \quad (3)$$

The equation to compute alarm trustworthiness is a bit different. If an alarm has different conditional trustwor-

thiness with different objects, the *Tru-Alarm* will take the maximum one as the result (Eq. (4)). Because even there is only one meaningful object that causes the alarm, the alarm is still meaningful. In Example 3,  $r_a(s_4, t)$  has higher trustworthiness when considering it being caused by object  $o_4$  rather than  $o_1$ . Even though  $o_1$  is found out to be non-existing, the alarm is still trustworthy since  $o_4$  is a meaningful object.

$$\tau(r_a(s, t)) = \max(\tau(r_a(s, t)|o)), o \in O_s \quad (4)$$

Algorithm 1 shows the framework to compute trustworthiness scores for the alarms and objects. For each object  $o$ , the system first retrieves its related data records from the object-alarm graph, and computes the conditional alarm trustworthiness (Lines 2–7). The object’s trustworthiness is then computed and the meaningful objects are selected out (Lines 8–9). The system scans object-alarm graph again and computes the trustworthiness for each alarm (Lines 11–15).

---

#### Algorithm 1: *Tru-Alarm* in a Snapshot

---

**Input:** Object-Alarm Graph  $G$  in snapshot  $t$ , trustworthiness threshold  $\delta_t$   
**Output:** Meaningful object set  $O_t$  and trustworthy alarm set  $Ra_t$

- 1 initialize  $O_t$  and  $Ra_t$ ;
- 2 **foreach** object  $o \in G$  **do**
- 3     retrieve monitoring sensor set  $S_o$  and related alarms  $Ra_o$ ;
- 4     **foreach** alarm  $r_a(s, t) \in Ra_o$  **do**
- 5         compute  $\tau(r_a(s, t)|o)$ ;
- 6         update corresponding edge weight in  $G$ ;
- 7     **end**
- 8     compute  $\tau(o)$ ;
- 9     if  $\tau(o) > \delta_t$  then add it to  $O_t$ ;
- 10 **end**
- 11 **foreach** alarm  $r_a(s, t) \in G$  **do**
- 12     retrieve the edges  $E$  linking to  $r_a(s, t)$ ;
- 13      $\tau(r_a(s, t)) = \max(E.weights)$ ;
- 14     if  $\tau(r_a(s, t)) > \delta_t$  then add it to  $Ra_t$ ;
- 15 **end**
- 16 **return**  $O_t$  and  $Ra_t$ ;

---

**Example 4.** Figure 4 shows a running example of the *Tru-Alarm* algorithm. For the 10 objects, the algorithm computes the conditional alarm trustworthiness, as listed in the first column. The object trustworthiness is then computed as the average of those scores in the second column. Object  $o_4$  is found as a meaningful one. Then the algorithm groups the conditional alarm trustworthiness by alarm and chooses the maximum one as the final score. The alarms reported by sensor  $s_4, s_5, s_7, s_8$  and  $s_9$  have high trustworthiness, the alarm by  $s_{15}$  is found as a false alarm.

#### IV. EFFICIENT TRUSTWORTHINESS INFERENCE

The time complexity of *Tru-Alarm* is linear in the number of objects. The efficiency will be a problem when there are a large number of objects generated by the localization algorithm. On the other hand, most objects turn out to be low trustworthy. In Example 4, there are 10 objects but  $o_4$  is found to be meaningful. *Tru-Alarm* wastes majority time on hopeless objects that do not really exist. The algorithm efficiency can be improved significantly if we can prune them in advance.

Conditional Alarm Trustworthiness (Group by Object)	Object Trustworthiness	Conditional Alarm Trustworthiness (Group by Sensor)	Alarm Trustworthiness
$\tau(r(s_4, t) o_1)=0.3$	$\tau(o_1)=0.15$	$\tau(r(s_4, t) o_1)=0.3$ <b><math>\tau(r(s_4, t) o_4)=0.92</math></b>	$\tau(r(s_4, t))=0.92$
$\tau(r(s_5, t) o_2)=0.27$	$\tau(o_2)=0.09$	$\tau(r(s_5, t) o_2)=0.27$ <b><math>\tau(r(s_5, t) o_4)=0.89</math></b> $\tau(r(s_5, t) o_3)=0.43$	$\tau(r(s_5, t))=0.89$
$\tau(r(s_8, t) o_3)=0.10$	$\tau(o_3)=0.05$	<b><math>\tau(r(s_7, t) o_4)=0.91</math></b> $\tau(r(s_7, t) o_5)=0.66$ $\tau(r(s_7, t) o_7)=0.59$ $\tau(r(s_7, t) o_8)=0.44$	$\tau(r(s_7, t))=0.91$
$\tau(r(s_4, t) o_4)=0.92$ $\tau(r(s_5, t) o_4)=0.89$ $\tau(r(s_7, t) o_4)=0.91$ $\tau(r(s_8, t) o_4)=0.82$ $\tau(r(s_9, t) o_4)=0.76$	$\tau(o_4)=0.86$	<b><math>\tau(r(s_8, t) o_4)=0.82</math></b> $\tau(r(s_8, t) o_3)=0.10$	$\tau(r(s_8, t))=0.82$
.....	.....	.....	.....
$\tau(r(s_{15}, t) o_9)=0.03$	$\tau(o_9)=0.01$	$\tau(r(s_{15}, t) o_9)=0.03$	$\tau(r(s_{15}, t))=0.04$
$\tau(r(s_{15}, t) o_{10})=0.04$	$\tau(o_{10})=0.02$	<b><math>\tau(r(s_{15}, t) o_{10})=0.04</math></b>	

Figure 4. Running Example of *Tru-Alarm*

**Property 1.** Let  $o$  be an object,  $S_o$  be its monitoring set and  $Ra_o$  be the set of related alarms.  $\tau(o)$ ’s upper-bound  $\overline{\tau(o)} = |Ra_o|/|S_o|$ .

**Proof:** Since the alarm trustworthiness’ value range is  $[0, 1]$ ,  $\tau(r_a(s, t)|o) < 1$ , then

$$\tau(o) = \frac{\sum_{r_a \in Ra_o} \tau(r_a(s, t)|o)}{|S_o|} < \frac{|Ra_o|}{|S_o|} \blacksquare$$

Intuitively, the meaningful objects are usually surrounded with many reliable alarming sensors, such as object  $o_4$  in Example 3. The objects with low trustworthiness often have some reliable sensors of normal readings nearby, like  $o_9$  and  $o_{10}$ .

Based on Property 1, we develop a *Tru-Alarm* with Object Pruning (TAOP) algorithm.

---

#### Algorithm 2: *Tru-Alarm* with Object Pruning

---

**Input:** Object-Alarm Graph  $G$  in snapshot  $t$ , trustworthiness threshold  $\delta_t$   
**Output:** Meaningful object set  $O_t$  and trustworthy alarm set  $Ra_t$

- 1 initialize  $O_t$  and  $Ra_t$ ;
- 2 **foreach** object  $o \in G$  **do**
- 3     retrieve monitoring sensor set  $S_o$  and related alarms  $Ra_o$ ;
- 4     compute  $\overline{p(o)}$ ;
- 5     if  $\overline{p(o)} < \delta_t$  then remove  $o$ ;
- 6     **foreach** alarm  $r_a(s, t) \in Ra_o$  **do**
- 7         compute  $\tau(r_a(s, t)|o)$ ;
- 8         update corresponding edge weight in  $G$ ;
- 9     **end**
- 10     compute  $\tau(o)$ ;
- 11     if  $\tau(o) > \delta_t$  then add it to  $O_t$ ;
- 12 **end**
- 13 **foreach** alarm  $r_a(s, t) \in G$  **do**
- 14     retrieve the edges  $E$  linking to  $r_a(s, t)$ ;
- 15      $\tau(r_a(s, t)) = \max(E.weights)$ ;
- 16     if  $\tau(r_a(s, t)) > \delta_t$  then add it to  $Ra_t$ ;
- 17 **end**
- 18 **return**  $O_t$  and  $Ra_t$ ;

---

Algorithm 2 computes  $\overline{\tau(o)}$  and prunes the hopeless

objects before carrying out trustworthiness inference (Lines 4 – 5). The remaining steps are the same as Algorithm 1. In the worst case, it cannot prune any objects beforehand, and the time complexity is the same as the original *Tru-Alarm*. However in our experiments more than 80% of the objects are pruned and the algorithm achieves a speed-up with an order of magnitude.

**Example 5.** Figure 5 shows a running example of TAOP algorithm. Before carrying out the trustworthiness inference process, the system first estimates the upper-bounds of each object. Suppose  $\delta_t$  is 0.4, then the objects  $o_2$ ,  $o_9$  and  $o_{10}$ 's upper-bounds are less than the threshold. They can be pruned without further computation. In another hand, there may be several false positives, e.g., object  $o_1$ 's upper-bound passes the threshold but its actually trustworthiness is still not qualified. TAOP algorithm checks each object after the inference steps and removes the false positives.

Object Trustworthiness Upperbound	Related Alarm Trustworthiness (Group by Object)	Object Trustworthiness
$\overline{\tau(o_1)} = 0.50$	$\tau(r(s_{4,t}) o_1)=0.3$	$\tau(o_1)=0.15$
$\overline{\tau(o_2)} = 0.33$ (pruned)	--	--
$\overline{\tau(o_3)} = 0.50$	$\tau(r(s_{8,t}) o_3)=0.10$	$\tau(o_3)=0.05$
$\overline{\tau(o_4)} = 1.00$	$\tau(r(s_{4,t}) o_4)=0.92$ $\tau(r(s_{5,t}) o_4)=0.89$ $\tau(r(s_{7,t}) o_4)=0.91$ $\tau(r(s_{8,t}) o_4)=0.82$ $\tau(r(s_{9,t}) o_4)=0.76$	$\tau(o_4)=0.86$
.....	.....	.....
$\overline{\tau(o_6)} = 0.25$ (pruned)	--	--
$\overline{\tau(o_8)} = 0.33$ (pruned)	--	--

Figure 5. Running Example of TAOP Algorithm

## V. PERFORMANCE EVALUATION

Three synthetic datasets are generated to test the performances of *Tru-Alarm*. The data generator simulates a battlefield with 600 to 2500 deployed sensors, tags the alarms caused by intrusion objects as trustworthy and intentionally adds many random noises to simulate the windblown debris and small animal activities. A summary of the datasets and the default parameter settings is listed in Figure 6.

The experiments are conducted on a PC with an Intel 2200 Quad CPU @ 2.66G Hz. The RAM is 8.19 GB and 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.

We first evaluate the time costs of the Original *Tru-Alarm* (OTA) and *Tru-Alarm* with Object Pruning (TAOP). Their running times are reported in Figure 7. Note that the y-axis is in logarithm scale. In all datasets, TAOP is an order of magnitude faster than OTA.

Dataset	Sensor#	Alarm#	True Alarm Rate
D1	625	5247	71%
D2	900	12390	46%
D3	2500	39415	29%
Parameter Settings			
Dataset: default D3			
Sampling Ratio %: default 4%			
$k$ in kNN: 4 to 16			

Figure 6. Experimental Settings

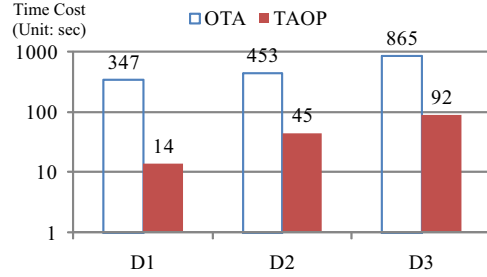


Figure 7. Time cost on Different Datasets

Two measurements are used in the studies of noise filtering and trustworthy alarm detection: (i)*Precision*: The proportion of trustworthy alarms over all detected alarms; (ii)*Recall*: The proportion of detected trustworthy alarms over all trustworthy alarms. If the system reports every alarm without any filtering, it achieves the best recall of 100%, but the precision is as low as the trustworthy alarm ratio of the dataset. This strategy is implemented as the baseline. We compare the algorithms of *Tru-Alarm* with kNN based methods in [3], parameter  $k$  is set to 4, 8 and 16. Figure 8 shows the precision and recall of evaluated methods with different trust threshold  $\delta_t$ . Since the object pruning technique does not influence the precision and recall, we omit the results of OTA in Figure 8.

Generally speaking, the TAOP has better precision than kNN based methods with reasonable  $\delta_t$ , and they can guarantee not missing any trustworthy alarms (achieve 100% recall). The kNN based methods have better precision in small dataset with high data qualities. However, the precision degenerates in the noisy datasets, because the kNN based methods judge an alarm's trustworthiness by comparing with its neighbor's readings. When the noise proportion is high, the sensor's neighbors may also report false alarms and kNN based methods will generate false positives. In the experiment kNN based methods always miss about 1% to 5% trustworthy alarms, which are reported by the *edge sensors* with normal reading neighbors. *Tru-Alarm* can easily find out those alarms by considering the object locations.

Another problem is that the trust threshold  $\delta_t$  is hard to tune for kNN based methods. A higher threshold helps achieve better precision but reduces the recall. Fortunately,

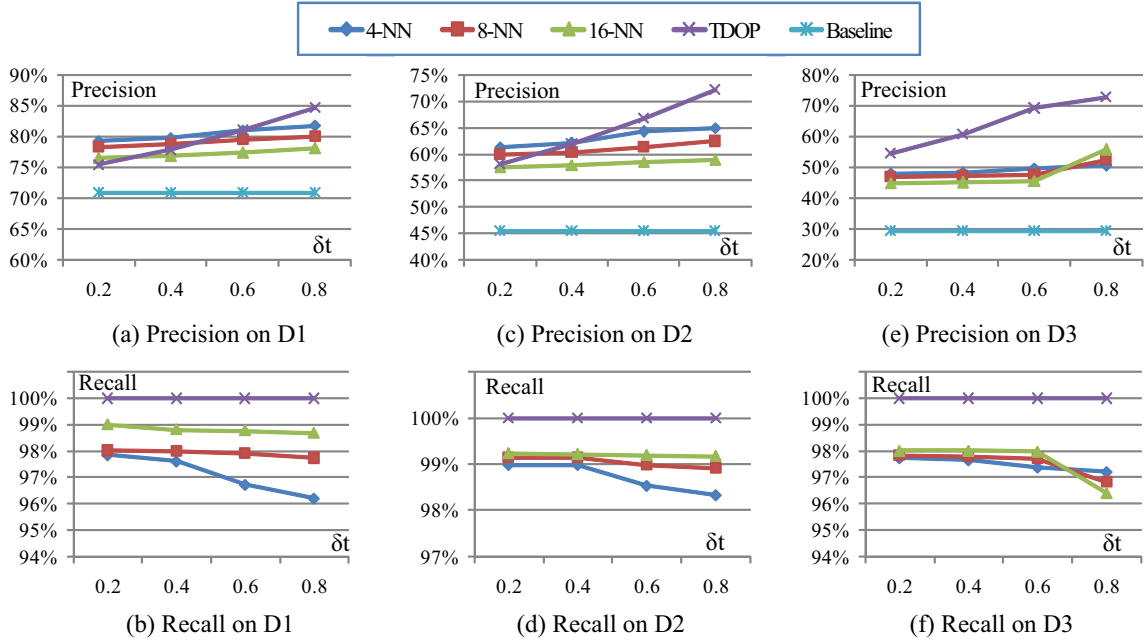


Figure 8. Results of Trustworthy Alarm Detection

*Tru-Alarm's* recall is robust to  $\delta_t$ . It keeps 100% even with a high threshold of 0.8. The users can safely tune  $\delta_t$  without worrying missing trustworthy alarms.

## VI. CONCLUSIONS

This paper studies the problem of trustworthiness analysis in cyber-physical systems. The authors propose the alarm and object trust models of sensor network. In the *Tru-Alarm* framework, the system constructs an object-alarm graph and carries out the trustworthiness inference along the links of such graph. Extensive experiments are conducted to show the scalability and applicability of proposed methods.

## VII. ACKNOWLEDGEMENTS

The work was supported in part by the NSF BDI-07-Movebank, NSF CNS-0931975, NSF IIS-10-17362, U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265, and by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA). Wen-Chih Peng was supported in part by the National Science Council, Project No. NSC 97-2221-E-009-053-MY3 and ATU program.

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 not withstanding any copyright notation here on.

## REFERENCES

- [1] <http://battlegroundiraq.wordpress.com/2007/11/19/thermal-sensor-protects-troops-boosts-security/>.
- [2] K. Baumgartner, S. Ferrari, and T. Wettergren. Robust deployment of dynamic sensor networks for cooperative track detection. In *IEEE Sensors Journal*, 2009.
- [3] X.-Y. Hsiao, W.-C. Peng, C.-C. Hung, and W.-C. Lee. Using sensorranks for in-network detection of faulty readings in wireless sensor networks. In *DEWMA*, 2007.
- [4] V. Isler, S. Kannan, and K. Daniilidis. Sampling based sensor-network deployment. In *IEEE RSJ*, 2004.
- [5] S. R. Jeffery, G. Alonso, Franklin, and Hong M. J. Declarative support for sensor data cleaning. In *ICPC*, 2006.
- [6] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. In *IEEE Trans. Comput.* 53, 3, 2004.
- [7] C. Lo and W. Peng. Carweb: A traffic data collection platform. In *MDM*, 2008.
- [8] K. Ni and G. Pottie. Bayesian selection of non-faulty sensors. In *IEEE International Symposium on Information Theory*, 2007.
- [9] PCAST. Supplement to the presidents budget for fiscal year 2008. In *The Networking and Information Technology Research and Development Program*, 2007.
- [10] L. Tang, B. Cui, and H. Li. Effective variation management for pseudo periodical streams. In *SIGMOD*, 2007.
- [11] L. Tang, B. Cui, and H. Li. Pgg: An online pattern based approach for stream variation management. *J. Comput. Sci. Technol.*, 23(4):497–515, 2008.
- [12] T. Wettergren. Performance of search via track-before-detect for distributed sensor networks. In *IEEE Trans. on Aerospace and Electronic systems*, 2008.
- [13] X. Yu, L. Tang, and J. Han. Filtering and refinement: A two-stage approach for efficient and effective anomaly detection. In *ICDM*, 2009.