Provenance-based Trustworthiness Assessment in Sensor Networks

Hyo-Sang Lim
Department of Computer Science,
Purdue University, USA
hslim@cs.purdue.edu

Yang-Sae Moon
Department of Computer Science,
Kangwon National University, South Korea
ysmoon@kangwon.ac.kr

Elisa Bertino
Department of Computer Science,
Purdue University, USA
bertino@cs.purdue.edu

ABSTRACT

As sensor networks are being increasingly deployed in decision-making infrastructures such as battlefield monitoring systems and SCADA(Supervisory Control and Data Acquisition) systems, making decision makers aware of the trustworthiness of the collected data is a crucial. To address this problem, we propose a systematic method for assessing the trustworthiness of data items. Our approach uses the data provenance as well as their values in computing trust scores, that is, quantitative measures of trustworthiness. To obtain trust scores, we propose a cyclic framework which well reflects the *inter-dependency* property: the trust score of the data affects the trust score of the network nodes that created and manipulated the data, and vice-versa. The trust scores of data items are computed from their *value similarity* and *provenance similarity*. The value similarity comes from the principle that "the more similar values for the same event, the higher the trust scores". The provenance similarity is based on the principle that "the more different data provenances with similar values, the higher the trust scores". Experimental results show that our approach provides a practical solution for trustworthiness assessment in sensor networks.

1. INTRODUCTION

Advances in hardware and network technologies enable the development of large-scale sensor networks in a large variety of novel applications, like supervisory systems, e-health, and e-surveillance. In near future, sensor networks will be deployed everywhere and consist of thousands to millions of tiny sensor nodes as we can see from the Smart Dust project [7] which aims to create grain-of-sand sized sensors. In such new environments, sensor networks collect large amounts of data that can convey important information for critical decision making. Thus, being able to assess the trustworthiness of the collected data and making decision makers aware of the trustworthiness of these data become crucial.

A possible approach to this problem is to associate a trust score with each data item. Such score provides an indication about the trustworthiness of the data item and can be used for data comparison or ranking. For example, even though the meaning of absolute scores varies depending on the application or parameter settings, if a data item has the highest trust score in a data set, then we can say that the data item is the most trustworthy compared with the other data items in the set. Also, as indicators about data trustworthiness, trust scores can be used together with other factors (e.g., information about contexts and situations, past data history) for deciding about the use of data items. A critical element in solutions to assign trust score to data is the method for computing the data trust scores. The goal of this paper is to develop such a method for data collected in sensor networks. Our approach is based on the concept of provenance, as provenance gives important evidence about the origin of the data, that is, where and how the data is generated. Provenance provides knowledge about how the data came to be in its current state - where the data originated, how it was generated, and the operations it has undergone since its creation.

Our method is based on the principle that the more trustworthy data a source provides, the more trusted the source is considered. There is thus an interdependency between network nodes and data items with respect to the assessment of their trust scores, i.e., the trust score of the data affects the trust score of the network nodes that created and manipulated the data, and vice-versa. To reflect such interdependency in computing trust scores, we propose a cyclic framework that generates: (1) trust scores of data items from those of network nodes and (2) trust scores of network nodes from those of data items. Trust scores are gradually evolved in our cyclic framework.

Our framework works as follows. Trust scores are initially computed based on the values and provenance of data items; we refer to these trust scores as *implicit trust scores*. To obtain these trust scores, we use two types of similarity functions: *value similarity* inferred from data values, and *provenance similarity* inferred from data provenances. Value similarity is based on the principle that the more data items referring to the same real-world event have similar values, the higher the trust scores of these items are. We thus propose a systematic approach for computing trust scores based on value similarity under the distribution of collected data. Provenance similarity is based on the observation that different provenances of similar data values may increase the trustworthiness of data items. In other words, different provenances provide more independent data items. In the paper we thus present a formal model for computing the provenance similarity and integrating it into the data similarity.

We have implemented the cyclic framework for computing trust scores. Through extensive experiments, we first show that our method works correctly in sensor networks and the cyclic framework gradually evolves trust scores by reflecting changes in sensing

DMSN'10, September 13, 2010, Singapore
.

value changes. These experimental results show that our approach provides a practical solution for trustworthiness assessment in sensor networks.

The rest of the paper is organized as follows. Section 2 models sensor networks and data provenances. Section 3 proposes the cyclic framework for generating trust scores of data items and network nodes based on their values and provenance. Section 4 reports the experimental results. We finally summarize and conclude the paper in Section 5.

2. DATA PROVENANCE AND ITS REPRESENTATION

Networks are usually modeled as graphs. We thus model the physical (sensor) network as a graph of $G(N, E)$, where the set of nodes, N, and the set of edges, E, are defined as follows:

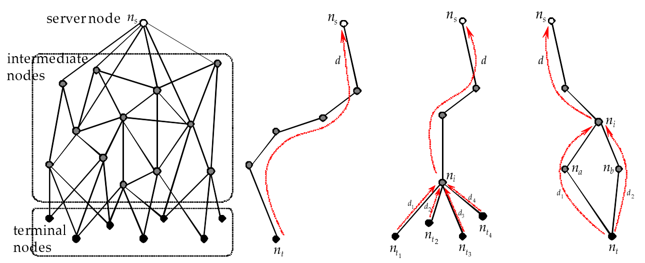
- $N = \{n_i \mid n_i$ is a network node of whose identifier is $i.\}$: a set of network nodes
- $E = \{e_{i,j} \mid e_{i,j}$ is an edge connecting nodes n_i and $n_j.\}$: a set of edges connecting nodes

Figure 1 (a) shows an example of a sensor network.

Regarding the network nodes in N, we categorize them into three types according to their roles.

Definition 1. *A terminal node generates a data item and sends it to one or more intermediate or server nodes. An intermediate node receives data items from one or more terminal or intermediate nodes, and it passes them to intermediate or server nodes; it may also generate an aggregated data item from the received data items and send the aggregated item to intermediate or server nodes. A server node receives data items and evaluates the user queries based on those items.* □

Without loss of generality, we assume that in G there is only one server node, denoted by n_s. To simplify the presentation, we only consider a single numeric value as a data item. However, we can easily extend our solution to multiple attributes by separately assigning independent scores to each attribute or by exploiting multi-attribute distributions. In this paper, we also only focus on handling selection and aggregation which are the most used operations in sensor networks. We will explore additional other operations in our future work.



(a) A physical sensor network (b) A simple provenance (c) An aggregate provenance (d) An exception

Figure 1: A physical sensor network and data provenance examples.

We now define the *provenance* of a data item d, denoted as p_d. The provenance p_d records where and how the data item d was generated and how it was passed to the server n_s.

Definition 2. *The provenance p_d of a data item d is a rooted tree satisfying the following properties: (1) p_d is a subgraph of the physical sensor network $G(N, E)$; (2) the root node of p_d is the server node n_s; (3) for two nodes n_i and n_j of p_d, n_i is a child of n_j if and only if n_i has passed the data item d to n_j.* □

We categorize intermediate nodes in data provenance into two types based on their operations. The *simple nodes* are internal node having only one child. The simple nodes simply pass data items from their children to their parents. Simple nodes are typically used in ad-hoc sensor networks to relay data items to a server in order to address the insufficient capability of data transmission. The *aggregate node* are internal nodes having two or more children nodes. They receive multiple data items from their multiple children, generate aggregated data items, and pass them to their parents.

Figures 1 (b) and 1 (c) show some examples of the two different data provenances. As shown in the figures, data provenances are subgraphs of the physical sensor network of Figure 1 (a), and they are trees rooted at the server node n_s. In Figure 1 (b) every intermediate node in the provenance p_d is a simple node, which means that the data item d is generated in a terminal node n_t and simply passed to the server n_s. We call this type provenance a *simple provenance*, which can be represented as a simple path. On the other hand, in Figure 1 (c) an internal node n_i is an aggregate node, which means that n_i generates a new data item d by aggregating multiple data items $d_1, \ldots, d_4$ from $n_{t_1}, \ldots, n_{t_4}$ and passes d to the server n_s. We call this type provenance an *aggregate provenance*, which is represented as a tree rather than a simple path.

According to Definition 2, a data provenance should be a tree. However, there could be cycles and thus the provenance is not a tree such as the example in Figure 1 (d). We do not consider this case because of two reasons. First, it rarely occurs in real environments. Second, tree similarity can be computed in $O(n^3 \log n)$ [6]; in contrast, computing graph similarity is known as an NP-hard problem [5] (refer to Section 3 for details). We note that basically there is no much difference between tree-shaped and graph-shaped provenances (Only minor changes are required to support graph-based provenance).

3. PROVENANCE-BASED TRUST SCORE COMPUTATION

In this section, we present our cyclic framework for computing trust scores of data items and network nodes.

3.1 Cyclic Framework for Incremental Update of Trust Scores

We derive our cyclic framework based on the *interdependency* [1, 3] between data items and their related network nodes. The interdependency means that the trust scores of data items affect the trust scores of network nodes, and similarly the trust scores of network nodes affect those of the data items. In addition, the trust scores need to be continuously evolved in the stream environment since new data items continuously arrive to the server. Thus, a cyclic framework is adequate to reflect the interdependency and continuous evolution properties. Figure 2 shows the cyclic framework according to which the trust scores of data items and the trust scores of network nodes are continuously updated. Note that we consider a sensor network where there are multiple sensors for monitoring an event (i.e., we can get multiple independent observations for an event), and thus trust scores are computed for the data items concerning the same event in a given streaming window.

As shown in Figure 2, we maintain three different types of trust scores, *current*, *intermediate*, and *next trust scores* to reflect the in-

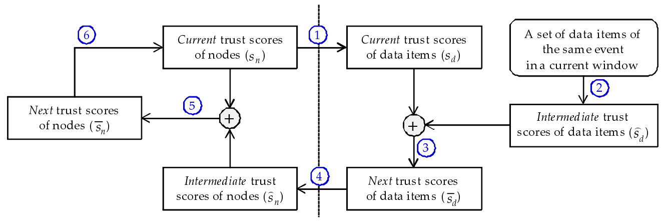

6
Current trust scores of nodes (s_n)
1
Current trust scores of data items (s_d)
A set of data items of the same event in a current window
2
Next trust scores of nodes ($\bar{s}_n$)
5
Intermediate trust scores of data items ($\hat{s}_d$)
3
Intermediate trust scores of nodes ($\hat{s}_n$)
4
Next trust scores of data items ($\bar{s}_d$)

Figure 2: A cyclic framework of computing trust scores of data items and network nodes.

terdependency and continuous evolution properties in the computation of the trust scores. The trust scores of data items and network nodes well reflect those properties as many as cycles are repeated. We explain the detailed computation process for the trust scores of data items and network nodes in Section 3.3 and 3.2, respectively.

It is important to note that these scores are mainly indicators, to be used for example for comparison purpose. For example, let s_1 and s_2 be trust scores of data d_1 and d_2. If $s_1 > s_2$, d_1 is more trustworthy than d_2. The meaning of absolute scores varies depending from the specific applications or parameter values.

3.2 Computing Trust Scores of Network Nodes

For a network node n whose current score is s_n, we are about to compute its next score $\bar{s}_n$. In more detail, the trust score of n was computed as s_n in the previous cycle, and we now recompute the trust score as $\bar{s}_n$ using a set of recent data items in a streaming window in order to determine how the trust score has to evolve in a new cycle. We compute the next score based on the following two principles: 1) the intermediate score $\hat{s}_n$ reflects the trust scores of its related data items based on the interdependency property; 2) the next score $\bar{s}_n$ reflects its current and intermediate scores s_n and $\hat{s}_n$ to gradually evolve the trust scores of network nodes.

We now show how to compute $\hat{s}_n$ and $\bar{s}_n$. First, let D_n be a set of data items that are issued from or passed through n in the given streaming window. That is, all data items in D_n are identified as related to the same event, and they are issued from or passed through the network node n. We adopt the idea that "higher scores for data items ($\in D_n$) result in higher scores for their related node (n)" [1, 2]. Thus, $\hat{s}_n$ is simply computed as the average of $\bar{s}_d$'s ($d \in D_n$), which are the next trust scores of data items in D_n:

$$\hat{s}_n = \frac{\sum_{d \in D_n} \bar{s}_d}{|D_n|} \tag{1}$$

In Eq. (1) we note that the trust score of a network node is determined by the trust scores of its related data items, and this satisfies the first principle, that is, interdependency property. Also, the next score $\bar{s}_n$ is computed as a weighted-sum of s_n and $\hat{s}_n$:

$$\bar{s}_n = c_n s_n + (1 - c_n)\hat{s}_n,$$
$$\text{where } c_n \text{ is a given constant of } 0 \le c_n \le 1. \tag{2}$$

In Eq. (2) we note that this satisfies the second principle, i.e., the consideration of current and intermediate scores.

The constant c_n in Eq. (2) represents how fast the trust score is evolved as the cycle is repeated. For example, if c_n has a larger value, especially if $c_n > \frac{1}{2}$, we consider s_n to be more important than $\hat{s}_n$, and this means that the previously accumulated historic score (s_n) is more important than the latest trust score ($\hat{s}_n$) recently computed from data items in D_n. On the other hand, if c_n has a smaller value, especially if $c_n < \frac{1}{2}$, we consider the latest score $\hat{s}_n$ to be more important than the historic score s_n. In summary, if c_n is large, the trust score will be evolved slowly; in contrast, if c_n is small, the trust score will be evolved fast.[1]

3.3 Computing Trust Scores of Data Items

Basically we compute the trust score of a data item d using its value v_d and provenance p_d. In this paper, we model the distributions of data items in the same event as a *normal (Gaussian) distribution*. In more detail, for data items in a set D of data items related to the same event, we model the distribution of D as a probability density function $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, where x is the attribute value v_d of a data item $d \, (\in D)$, and μ and σ^2 are mean and variance of D respectively. We use the normal distribution since it well reflects natural phenomena. Especially, values sensed for one purpose in general follow a normal distribution [4, 8], and thus this distribution is a reasonable choice for modeling streaming data items in sensor networks. However, we note that the normal distribution assumption is not a limit of our solution. We just use the distribution for estimating similarities among data values in the trust score computation. We can adopt other distributions, histograms, or correlation information with simple changes to the data similarity models.

3.3.1 Current trust score s_d

For a data item d, we first compute its current score s_d based on current scores of network nodes in its provenance p_d (see ① in Figure 2). This process reflects the interdependency property because we use the trust scores of network nodes for those of data items. In Section 2 we have explained the two different types of provenance: one is the simple provenance with a path structure; the other is the aggregate provenance with a tree structure. According to this classification, in what follows we first show how to compute the current score s_d for the simple provenance and then extend it for the aggregate provenance.

In the case of the simple provenance (like in Figure 1 (b)), we can represent it as $p_d = (n_1, n_2, \ldots, n_k = n_s)$, that is, as a sequence of network nodes that d passes through. In this case, we determine the current score s_d on the minimum among the scores of all nodes in p_d. This is based on an intuition that, if a data item passes through network nodes in a sequential order, its trust score might be dominated by the worst node with the smallest trust score[2]. That is, we compute s_d as follows:

$$s_d = \min\{s_{n_i} \,|\, n_i \in p_d\} \tag{3}$$

If a data item d has an aggregate provenance p_d, we need to consider the tree structure (like in Figure 1 (c)) to compute its current score s_d. For an aggregate node, we first obtain a representative score by aggregating the current scores of its child nodes and then use this aggregate score as the current score of the child nodes. We use an average score of child nodes as their aggregated score[3]. By recursively executing this aggregation process, we simplify a tree into a simple path of aggregated scores, and we finally compute the current score s_d by taking their minimum score as in Eq. (3).

Algorithm 1 shows a recursive solution for computing the current score s_d from its provenance p_d, which can be either a simple or

[1]In the experiment we set $c_n = \frac{1}{2}$ to equally reflect the importance of s_n and $\hat{s}_n$, and we assume that the first value of s_n is set to 1.

[2]We can also use an average score or weighted average score of network nodes to compute the current score. In this case, we obtain the score by simply changing the minimum function to the average or weighted average function in Eq. (3).

[3]According to the aggregate operation applied to the aggregate node, we can use different methods. That is, for `AVG` we can use an average of children, but for `MIN` or `MAX` we can use a specific score of a network node that produces a resulting minimum or maximum value. An aggregation itself, however, represents multiple nodes, and we thus use the average score of child nodes as their representative score.

aggregate provenance. To obtain the current score s_d of a data i
d, we simply call *CompCurrentScore*(n_s) where n_s is the root n
of p_d.

Algorithm 1 *CompCurrentScore* (n_i: a tree node in p_d)

```
1:  if $n_i$ is a simple node (i.e., $n_i$ has only one child) then
2:      Let $n_j$ be the child node of $n_i$;  // an edge $e_{i,j}$ connects two nodes.
3:      return MIN($s_{n_i}$, CompCurrentScore($n_j$));
4:  else if $n_i$ is an aggregate node with $k$ children then
5:      Let $n_{j_1}, \ldots, n_{j_k}$ be $k$ child nodes of $n_i$;
6:      return MIN($s_{n_i}$, AVG(CompCurrentScore($n_{j_1}$), . . .,
7:                         CompCurrentScore($n_{j_k}$)));
8:  else  // $n_i$ is a leaf node.
9:      return $s_{n_i}$;
10: end-if
```

3.3.2 Intermediate trust score $\hat{s}_d$

An intermediate trust score $\hat{s}_d$ of a data item d is computed f
the latest set of data items of the same event with d in the cur
streaming window (see ② in Figure 2). Let D be the set of
items in the same event with d. In general, if set D changes,
a new item is added to D or an item is deleted from D, we rec
pute the trust scores of the data items in D. We obtain $\hat{s}_d$ thro
the initial and adjusting steps. In the initial step, we use the *va*
similarity of data items in computing an initial value of $\hat{s}_d$. In
adjusting step, we use the *provenance similarity* to adjust the ini
value of $\hat{s}_d$ by considering provenances of data items.

(1) Initial score of $\hat{s}_d$ based on value similarity

Based on our normal distribution model, we observe that, for a
D of a single event, its mean μ is the most representative va
that well reflects the value similarity. This is because the mea
determined by the majority values, and obviously those majo
values are similar to the mean in the normal distribution. Thus,
conclude that the mean has the highest trust score; if the value
a data item is close to the mean, its trust score is relatively high
the value is far from the mean, its trust score is relatively low.

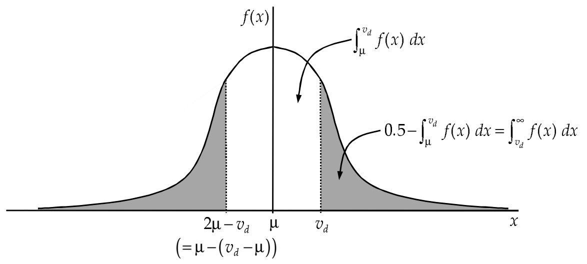
Based on those observations, we propose a method to compute the intermediate score s_d in the initial step. In obtaining $\hat{s}_d$, we assume that $v_d \geq \mu$. We can easily extend our method to the case of $v_d \leq \mu$, and we thus omit that case for simplicity.

As intermediate score $\hat{s}_d$, we use the cumulative probability of the normal distribution. In this method, we use "1 − the amount of how far v_d is from the mean" as the initial score of $\hat{s}_d$, and here "the amount of how far v_d is from the mean" can be thought as the cumulative probability of v_d. Thus, as in Eq. (4), we obtain the initial $\hat{s}_d$ as the integral area of $f(x)$.

$$\begin{aligned}\hat{s}_d &= 2\left(0.5 - \int_{\mu}^{v_d} f(x)\,dx\right)\\ &= 1 - \int_{2\mu - v_d}^{v_d} f(x)\,dx = 2\int_{v_d}^{\infty} f(x)\,dx \qquad (4)\end{aligned}$$

Figure 3 shows how to compute the integral area for the initial intermediate score s_d. In the figure, the shaded area represents the initial score of $\hat{s}_d$, which is obviously in (0,1]. Here, the score $\hat{s}_d$ increases as v_d is close to μ.

According to our data similarity model, if a sensor suddenly generates a data value which is different from the mean, this data value will initially receive a low trust score. However, in this case, the system provides users with an explanation about why the trust score is so low. There could be two possible reasons for the trust score of a data generated by a sensor to be low: 1) the observation by the sensor is quite different from the other observations of the same event; 2) the observation is currently the only observation for the event. In the former case, users can safely conclude that the data value is wrong. In the latter case, users can take different actions. They can just wait for the arrival of new data concerning this event. Or they can activate additional sensors (for example we may assume that not all sensors are always activated in order to save energy), that in turn will result in more data to be generated. If the initial observation is actually true, other sensors will send similar observations shortly, and then, the cyclic framework will automatically reflect this situation by increasing the trust scores of the initial data value.



$f(x)$
$\int_{\mu}^{v_d} f(x)\,dx$
$0.5 - \int_{\mu}^{v_d} f(x)\,dx = \int_{v_d}^{\infty} f(x)\,dx$
$2\mu - v_d$
μ
v_d
x
$(=\mu-(v_d-\mu))$

Figure 3: Computing the intermediate score of $\hat{s}_d$.

(2) Adjusted score of $\hat{s}_d$ based on provenance similarity

We need to adjust the intermediate score $\hat{s}_d$ by reflecting the provenance similarity of data items. To achieve this, we let a set of provenances in D be P and the similarity function between two data provenances p_i, p_j ($\in P$) be $sim(p_i, p_j)$[4]. Here, the similarity function $sim(p_i, p_j)$ returns a similarity value in $[0, 1]$, and can be computed from the tree or graph similarity [5, 6]. Computing graph similarity, however, is known to be an NP-hard problem [5], and we thus use the tree similarity [6], which is an edit distance-based similarity measure.

Our approach to take into account provenance similarity in computing the intermediate score $\hat{s}_d$ is based on some intuitive observations. In the following, notation '$\sim$' means "is similar to", and notation $\nsim$ means "is not similar to." Given two data items $d, t \in D$, their values v_d, v_t, and their provenances $p_d, p_t \in P$,

- if $p_d \sim p_t$ and $v_d \sim v_t$, the provenance similarity makes a *small positive* effect on $\hat{s}_d$;
- if $p_d \sim p_t$ and $v_d \nsim v_t$, the provenance similarity makes a *large negative* effect on $\hat{s}_d$;
- if $p_d \nsim p_t$ and $v_d \sim v_t$, the provenance similarity makes a *large positive* effect on $\hat{s}_d$;
- if $p_d \nsim p_t$ and $v_d \nsim v_t$, the provenance similarity makes a *small positive* effect on $\hat{s}_d$;

Based on the above observations, we introduce a measure of *adjustable similarity* to reflect the provenance similarity in adjusting $\hat{s}_d$. Given two data items d, t ($\in D$), we first define the adjustable similarity between d and t, denoted by $\rho_{d,t}$, as follows:

$$\rho_{d,t} = \begin{cases} 1 - sim(p_d, p_t), & \text{if } dist(v_d, v_t) < \delta_1; \text{// positive effect} \\ -sim(p_d, p_t), & \text{if } dist(v_d, v_t) > \delta_2; \text{// negative effect} \\ 0, & \text{otherwise. // no effect} \end{cases} \qquad (5)$$

In Eq. (5), $dist(v_d, v_t)$ is a distance function between v_d and v_t; δ_1 is a threshold indicating when v_d and v_t are to be treated as similar; δ_2 is a threshold indicating when v_d and v_t are to be treated as

[4]Data items in the same event may have similar provenances, so we may assume that the number of possible provenances for an event is finite and actually small. Thus, for real-time processing purposes, we can materialize all $sim(p_i, p_j)$'s in advance and maintain them in memory.

dissimilar[5]. The adjustable similarity $\rho_{d,t}$ in Eq. (5) well reflects the effect of provenance and value similarities. That is, if v_d and v_t are similar, $\rho_{d,t}$ has a positive value of "$1 - sim(p_d, p_t)$" determined by the provenance similarity; in contrast, if they are not similar, $\rho_{d,t}$ has a negative value of "$-sim(p_d, p_t)$." To consider adjustable similarities of all data items in D, we now obtain their sum ρ_d as follows:

$$\rho_d = \sum_{t \in D, t \neq d} \rho_{d,t} \tag{6}$$

We then adjust the value v_d by considering ρ_d and use the adjusted value, denoted by $\bar{v}_d$, to compute $\hat{s}_d$ instead of v_d. In more detail, we first normalize ρ_d into $[-1, 1]$ using its maximum and minimum similarities, $\rho_{\max}$ and $\rho_{\min}$. The normalized value of ρ_d, denoted by $\bar{\rho}_d$, is thus computed as follows:

$$\bar{\rho}_d = 2\frac{\rho_d - \rho_{\min}}{\rho_{\max} - \rho_{\min}} - 1, \text{ where } \rho_{\max} = \max\{\rho_t | \, t \in D\}$$
$$\text{and } \rho_{\min} = \min\{\rho_t | \, t \in D\} \tag{7}$$

We then adjust the data value v_d to a new value $\bar{v}_d$ as follows:

$$\bar{v}_d = \min\{v_d - \bar{\rho}_d(c_p \cdot \sigma), \mu\},$$
$$\text{where } c_p \text{ is a constant greater than 0.} \tag{8}$$

Figure 4 shows how the value v_d changes to $\bar{v}_d$ based on the adjustable similarity $\bar{\rho}_d$ in the framework of a normal distribution. As shown in the figure, if $\bar{\rho}_d > 0$, i.e., if the provenance similarity makes a positive effect, v_d moves to the left in the distribution graph, i.e., the intermediate score $\hat{s}_d$ increases; in contrast, if $\bar{\rho}_d < 0$, i.e., if the provenance similarity makes a negative effect, v_d moves to the right in the graph, i.e., $\hat{s}_d$ decreases. In Eq. (8), c_p represents the important factor of provenance similarity in computing the intermediate score. That is, as c_p increases, the provenance similarity becomes more important. We use 0.2 as the default value of c_p, i.e., we move the data value v_d in $\pm 20\%$ range of the standard deviation σ.

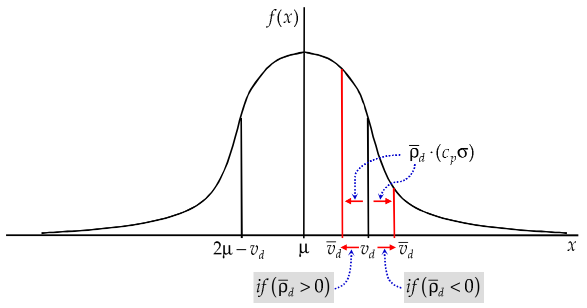


Figure 4: The effect of the provenance similarity on a data value.

By using the adjusted data value $\bar{v}_d$, we finally recompute the intermediate score $\hat{s}_d$. By simply changing v_d to $\bar{v}_d$, we can also obtain Eq. (9) from Eq. (4) in which the integral area for the intermediate score may increase or decrease by the provenance similarity.

$$\hat{s}_d = 2\int_{\bar{v}_d}^{\infty} f(x)\,dx = 1 - \int_{2\mu - \bar{v}_d}^{\bar{v}_d} f(x)\,dx \tag{9}$$

[5]In the experiment we set δ_1 and δ_1 to 20% and 80% of the average distance, respectively.

3.3.3 *Next trust score* $\bar{s}_d$

For a data item d we eventually compute its next trust score $\bar{s}_d$ by using the current score s_d and the intermediate score $\hat{s}_d$. In obtaining $\bar{s}_d$, we use s_d for the interdependency property since s_d is computed from network nodes, and we exploit $\hat{s}_d$ for the continuous evolution property since $\hat{s}_d$ is obtained from the latest set of data items. Similar to computing the next score $\bar{s}_n$ of a network node n in Eq. (2), we compute $\bar{s}_d$ as follows:

$$\bar{s}_d = c_d s_d + (1 - c_d)\hat{s}_d,$$
$$\text{where } c_d \text{ is a given constant of } 0 \leq c_d \leq 1. \tag{10}$$

As shown in Eq. (10), the next score $\bar{s}_d$ is gradually evolved from the current and intermediate scores s_d and $\hat{s}_d$. We also note that $\bar{s}_d$ will be used to compute the intermediate scores (i.e., $\hat{s}_n$) of network nodes in the next computation cycle (see ④ in Figure 2) for the interdependency and continuous evolution principles.

Like constant c_n used in computing s_n for a network node n in Eq. (2), constant c_d in Eq. (10) represents how fast the trust score evolves as the cycle is repeated. If c_d is large, the trust scores of data items evolve slowly; in contrast, if c_d is small, they evolve fast.[6]

In this section, instead of calibrating our model with real data sets, we present general principles for choosing parameter values (e.g., confidence ranges control the tradeoff between the number and quality of results, c_n controls how fast scores are evolved). We believe these principles can be used in most applications.

4. EXPERIMENTAL EVALUATION

In this section, we present our performance evaluation. In what follows, we first describe the experimental environment, and then present the experimental results.

4.1 Experimental Environment

The goal of our experiments is to evaluate the *efficiency* and *effectiveness* of our approach for the computation of trust scores. To evaluate the efficiency, we measure the elapsed time for processing a data item with our cyclic framework in the context of a large scale sensor network and a large number of data items. To evaluate the effectiveness, we simulate an injection of incorrect data items into the network and show that trust scores rapidly reflect this situation.

We simulate a sensor network for the experiments. For simplicity, we model our sensor network as an f-ary complete tree whose fanout and depth are f and h, respectively. We vary the values of f and h to control the size of sensor networks for assessing the scalability of our framework. We also set the number of unique events to N_{event}.

We use synthetic data that has a single attribute whose values follow a normal distribution with mean μ_i and variance ${\sigma_i}^2$ for each event $i\,(1 \leq i \leq N_{event})$. To generate data items, for each event, we assign N_{assign} leaf nodes of the sensor network with an interleaving factor $N_{interleave}$. This means that the data items for an event are generated at N_{assign} leaf nodes and the interval between the assigned nodes is $N_{interleave}$ (e.g., if $N_{interleave} = 0$, then N_{assign} nodes are exactly adjacent with each other). To simulate the incorrect data injection, we randomly choose an event and a node assigned for the event, and then, generate a random value.

For computing the similarity between two provenances p_i and p_j (i.e., *sim*(p_i,p_j)), we use a path edit distance defined as follows:

$$sim(p_i, p_j) = 1 - \frac{1}{h}\sum_{k=1}^{h}\frac{\text{node distance between } p_i \text{ and } p_j \text{ at the } k\text{-th level}}{\text{total number of nodes at the } k\text{-th level}}$$

[6]In the experiment we set $c_d = \frac{1}{2}$ to equally reflect the importance of s_d and $\hat{s}_d$.

Here, the node distance is defined as the number of nodes between two nodes at the same level.

All the experiments have been conducted on a PC with a 2.2GHz Core2 Duo processor and 2GB RAM running Windows/XP. The program code has been written in Java with JDK 1.6.0. Table 1 summarizes the experimental parameters and their default values. In all experiments we use the default values unless mentioned otherwise.

Table 1: Summary of notation.

Symbols	Definitions	Default
h	height of the sensor network	5
f	fanout of the sensor network	8
N_{event}	# of unique events	1000
N_{assign}	# of nodes assigned for an event	30
$N_{interleave}$	interleaving factor	1
ω	size of window for each event	20

As can be seen in Table 1 we only vary some application insensitive parameters. The other parameters (e.g., weights, thresholds) may be more sensitive to application contexts (e.g., data distributions, attack patterns). We will consider these parameters in the context of specific applications in our future work.

4.2 Experimental Results

(1) *Computation efficiency*: We measured the elapsed time for processing a data item. Figure 5 reports the elapsed times for different values of h's and ω's.

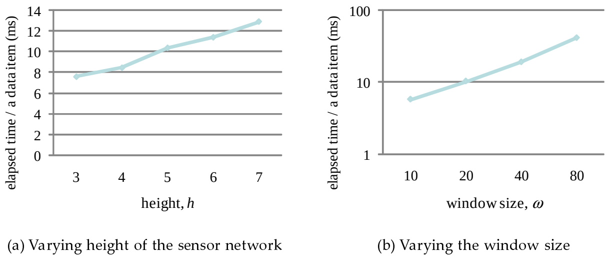


Figure 5: Elapsed times for computing trust scores.

From Figure 5 (a), we can see that the elapsed time increases as h increases. The reason is that, as h increases, the length of provenance also increases. However, the increasing rate is not high; for example, the elapsed time increases only by 9.7% as h varies from 5 to 6. The reason is that the additional operations for longer provenance linearly increase when computing the trust scores for both data items and network nodes. For the data items, only s_d and $\hat{s}_d$ are affected by the length of the provenance, i.e., an additional iteration is required to compute the weighted sum for s_d and a provenance similarity comparison for $\hat{s}_d$. For the network nodes, the computation cost increases linearly with the height (not with the total number of nodes), since we consider a very small number of network nodes related to the provenance of the new data item.

From Figure 5(b), we can see that the elapsed time increases more sharply as ω increases. The reason is that the number of similarity comparisons (not an iteration) for $\hat{s}_d$ linearly increases as ω increases. However, we can see that the performance is still adequate for handling high data input rates; for example, when ω is 80, the system can process 25 data items per second.

(2) *Effectiveness*: To assess the effectiveness of our approach, we injected incorrect data items into the sensor network, and then observed the change of trust scores of data items. Figure 6 shows the trend in trust score changes for different values of the interleaving factor $N_{interleave}$. Here, $N_{interleave}$ affects the similarity of provenances for an event, i.e., if $N_{interleave}$ increases, the provenance similarity decreases.

Figure 6 (a) shows the changes in the trust scores when incorrect data items are injected. The figure shows that the trust scores change more rapidly when $N_{interleave}$ is smaller. This trend is explained by the principle "different values with similar provenance result in a large negative effect." In contrast, Figure 6 (b) shows the changes when the correct data items are generated again. In this case, we can see that the trust scores are modified more rapidly when $N_{interleave}$ is larger. This trend is explained by the principle "similar values with different provenance result in a large positive effect."

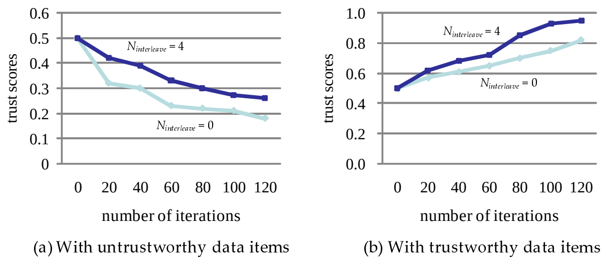


Figure 6: Changes of trust scores for incorrect data items.

5. CONCLUSIONS

In this paper we propose a systematic method for computing and evolving the trustworthiness levels of data items/network nodes. We first introduce a cyclic framework of computing actual trust scores of data items and network nodes based on the interdependency between data and network nodes. We then provide a formal method for computing trust scores based on the *value* and *provenance similarities* of data items. Through extensive experiments, we show that our cyclic framework works well in sensor networks.

As future work, we plan to: (1) consider multiple *dependent* attributes and multi-attributes in-network operations and (2) consider other probability distributions instead of normal distributions.

Acknowledgements: The work of Elisa Bertino and Hyo-Sang Lim has been partially supported by Northrop Grumman as part of the NGIT Cybersecurity Research Consortium and by the NSF Grant N.0964294 "NeTS: Medium: Collaborative Research: A Comprehensive Approach for Data Quality and Provenance in Sensor Networks".

6. REFERENCES

[1] E. Bertino, C. Dai, H.-S. Lim, and D. Lin, "High-Assurance Integrity Techniques for Databases," In *Proc. of the 25th British Nat'l Conf. on Databases*, Cardiff, UK, pp. 244-256, July 2008.

[2] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu, "An Approach to Evaluate Data Trustworthiness Based on Data Provenance," In *Proc. of the 5th VLDB Workshop on Secure Data Management*, Auckland, New Zealand, pp. 82-98, Aug. 2008.

[3] C. Dai et al., "Query Processing Techniques for Compliance with Data Confidence Policies," In *Proc. of the 6th VLDB Workshop on Secure Data Management*, Lyon, France, pp. 49-67, 2009.

[4] E. Elnahrawy and B. Nath, "Cleaning and Querying Noisy Sensors," In *Proc. of the 2nd ACM Int'l Conf. on Wireless Sensor Networks and Applications*, San Diego, California, pp. 78-87, Sept. 2003.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, 1990.

[6] P. N. Klein, "Computing the Edit-distance between Unrooted Ordered Trees," In *Proc. of the 6th Annual European Symposium on Algorithms (ESA)*, Venice, Italy, pp 91-102, Aug. 1998.

[7] Smart Dust Project, http://robotics.eecs.berkeley.edu /∼pister/SmartDust/.

[8] M. Rabbat and R. Nowak, "Distributed Optimization in Sensor Networks," In *Proc. of the 3rd Int'l Symp. on Information Processing in Sensor Networks*, Berkeley, California, pp. 20-27, Apr. 2004.