Fog-Based Evaluation Approach for Trustworthy Communication in Sensor-Cloud System


Abstract—A sensor-cloud system is a combination of wireless sensor networks (WSNs) and cloud computing that is equipped with ubiquitous physical sensing ability, high-speed computation, huge storage, and so on. However, sensor-cloud systems suffer from various types of malicious attacks that can cause sensor communications to become unreliable. Establishing a trust evaluation method to ensure members’ reliability in sensor-cloud is an effective way to resist malicious attacks. However, most current trust evaluation methods are constrained to specific attacks or applications, and they lack compatibility, verifiability and scalability. To solve these problems, we formulated the trust evaluation issue as a multiple linear regression problem. Considering energy restrictions, we adopt fog nodes to assist in the trust computation. Moreover, the least squares algorithm is used to find the fitting function between the communication feature and the trust value. The experimental results show that our approach can find the best trust evaluation model and improve the compatibility, verifiability and accuracy of trust evaluation.

Index Terms—sensor-cloud system, security, trust, multiple linear regression, least squares.

I. INTRODUCTION

SENSOR-cloud system is a combination of wireless sensor networks (WSNs) and cloud computing [1]. Traditional WSNs have limited power, storage and processing capabilities, limiting the fields to which they can be applied, such as Z. Jian et al [2] adopted mobile sink to improve the throughput of WSNs. Merging WSNs with cloud computing strengthens their performance by extending their storage capacity, processing complexity, and so on [3]. In a sensor-cloud system, the sensors collect physical data and upload it to cloud for storage and processing. Then, the cloud servers can not only serve to store and process the data, but can also be used to extract useful information to provide services [4]. Related technologies are developed at same time, such as Shen et al [5] propose an efficient public auditing protocol to reduce computational and communication overheads. In this way, the public sensed data can be used by more people not only the WSNs owner. For example, S. Jian et al [6] propose a cloud-assisted urban data sharing framework, which furnishes human a good condition to build ubiquitous-cities.

However, the wireless communication and resource-constrained nature of WSNs are also convenient for malicious attackers who wish to disrupt sensor-cloud communications [7]. Fu et al [9] employ the multi-keyword technology over encrypted cloud data to solve the problem of privacy-preserving. Establishing a trustworthy communication between sensors is a necessary mechanism to ensure that all the sensors are trusted. Various types of trust models in the current literatures can be divided into four categories: risk factor, belief, subjective probability and transitivity relationship. Zhang T. et al. [12], proposed taking additional parameters into account, such as historic communication records, transmission distance, energy and so on. However, three serious problems remain to be solved: 1) compatibility, the definition of trust varies between studies; 2) verifiability, there is no universal standard to verify the effectiveness of trust models; and 3) accuracy, the limited energy and processing ability of sensors restricts trust model designs, which, in turn, affects trust evaluation accuracy. To solve these problems, we establish an elastic trust model by formulating the trust evaluation issue as a multiple linear regression problem based on features of malicious attacks. Moreover, we employ fog nodes to help with the trust computations.

First, establishing the relationship between sensor’s behavior and its trust value is similar to a multiple linear regression problem. Therefore, the least squares algorithm can be used to find the best fitness function based on malicious attack features. One remarkable fact is that the attack features for trust evaluation are not fixed; They can be supplemented or reduced based on security requirements. Second, a fog node is a type of device with strong capabilities, such as large capacity and fast computing power that bring virtual cloud services to the edge of the network. All fog nodes are implemented as intermediate platform between end devices and cloud computing data centers [13].

In this paper, we first establish a universal trust model based on the sensors’ communication history feature sets. Then, when sensors upload communication features to fog nodes during each upload interval, the fog nodes compute the trust value of the communications and divide the sensors into three classes: credible nodes, suspect nodes and unbelievable nodes. If a sensor is judged as an unbelievable node, the fog node broadcasts that information to the other sensors. Subsequently, the unbelievable node cannot gain access to resources, such as channels, data, and so on. The main contributions of this paper are as follows:

1. We find three significant problems in the current trust model for sensor-cloud systems and design a fog-based trust evaluation approach to solve these problems. Our approach finds the best fitting function and improves the trust evaluation accuracy.
2. We formulate the trust evaluation issue as a multiple linear regression problem and establish the trust model based on attack features. The authenticity of trust value can be verified and this approach also solves the compatibility problem.

II. OVERVIEWS

A. Multiple Linear Regression Model

To establish the notation used later in this paper, we will use $x^{(i)}$ to denote the "input" variables, also called input features (e.g., energy or radius), and $y^{(i)}$ to denote the "output" or trust value that we are trying to evaluate. A pair of such examples $(x^{(i)}, y^{(i)})$ is called a training example, and the dataset consists of a list of $m$ training examples $(x^{(i)}, y^{(i)}); i = 1, 2, ..., m$ also called the training set. Note that the superscript "$^i$" in the notation is simply an index into the training set; it does not denote exponentiation. We also use $X$ to denote the set of input values, and $Y$ to denote the set of output values. Given a training set, a function $H : X \mapsto Y$ is determined such that $H(x)$ is "good" predictor for the corresponding value of $y$.

For historical reasons, the function $H$ is called a hypothesis. In this paper, to apply the linear regression function to trust evaluation, we first represent $H$ using a computer. As an initial choice, assume we decide to approximate $y$ as a linear function of $x$: $H_0(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$. Here, the $\theta_i$'s are parameters (also called weights) that parameterize the linear function mapping space from $X$ to $Y$. To simplify our notation, we note $H_0(x)$ as $H(x)$, and introduce the convention of letting $x_0 = 1$; therefore, $H(x) = \sum^n_{i=0} \theta_i x_i = \theta^T x$. On the right-hand side of the preceding equation, we consider that $\theta$ and $x$ are both vectors, and $n$ is the number of input variables (not counting $x_0$). To make $H(x)$ close to $y$ for our training set, we must determine a function that measures, for each $\theta$ value, how close the $H(x^{(i)})$'s are to the corresponding $y^{(i)}$'s. We define the cost function: $J(\theta) = \frac{1}{2} \sum^m_{i=1} (H_0(x^{(i)}) - y^{(i)})^2$. The solution for finding the minimum value of the cost function will be given in Section 4.

B. Feature Model

From the attack model description, we determine some useful features that can be used as input variables (see Table I). These features are divided into two categories: $c_1$ contains some useful features but does not take part in trust computation, while $x_i$ is the parameter to compute the sensors’ trust value. Each sensor is equipped with a unique ID. When the sensors communicate with the fog nodes, this ID is attached to the package. The routing information and the sensor position also inform cloud servers, because they can assist the cloud in judging a sensors’ realistic communication radius and determine whether the sensor is a newcomer. The package forwarding rate is an important factor in all types of attacks. Moreover, a realistic communication radius is the basic parameter for detecting replicated attacks. Beyond that, the feature, response time and energy consumption are related to all attacks. Note that these features are not absolute for trust evaluation and can be supplemented based on more security requirements.

### Table I: Feature Model

<table>
<thead>
<tr>
<th>Features</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>Unique ID of each sensor</td>
</tr>
<tr>
<td>$c_2$</td>
<td>List of communicating objects</td>
</tr>
<tr>
<td>$c_3$</td>
<td>Position information</td>
</tr>
<tr>
<td>$c_4$</td>
<td>Max delivery distance</td>
</tr>
<tr>
<td>$x_1$</td>
<td>Package delivery ratio</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Response time</td>
</tr>
<tr>
<td>$x_3$</td>
<td>Average energy consumption</td>
</tr>
</tbody>
</table>

C. Fog-based Structure

Though sensor-cloud systems include the capacities of both WSNs and cloud computing, there are still some weaknesses, including delivery delays, response times, and so on. These features should be considered when designing a trust evaluation algorithm. Therefore, the fog layer is introduced to assist the physical layer by conducting complicated computations and providing faster feedback to sensors. The fog layer is implemented as an intermediate platform between the physical layer and the cloud. Sensors transmit the feature set to a fog node for trust evaluation. The fog nodes compute the trust value based on the feature set and return the result to sensors. Additionally, information exchange between the fog layer and the cloud is more efficient than that in a typical sensor-cloud system.

III. FOG-BASED TRUST COMPUTATION

After having introduced the two primary aspects of our approach, we present the detailed design of the fog-based trust computation in this section.

A. Trust Evaluation Process

The fog-based trust evaluation process is shown in Figure 1. Before beginning the evaluation, we first construct feature training datasets to establish trust model. The training dataset contains two parts: a feature set and a corresponding trust value set. One group feature corresponds to one trust value. Then, a least squares algorithm is used to find an optimal set of values of $\theta$ that minimize the cost function $J(\theta)$. When a $\theta$ is confirmed, the fitness function $H_0(X)$ is complete. Subsequently, when a sensor uploads its feature data to a fog node, the fog node first checks the routing information to determine whether there the node has a new neighbor ID. If the result is yes, then the fog node checks the authenticity of new sensor. When this sensor has a legal identity and its ID does not duplicate others, its trust value will be computed by $H(\theta)$ based on the feature; otherwise, its trust value is zero. There are four conditional judgments in the process. For the condition $C_1$ and $C_2$, when a new sensor joins the Internet, the fog node first checks all the existing sensors to compare the new ID and check its identity to protect systems from Sybil attacks. For the condition $C_3$, checking for duplicate IDs can resist replication attacks. Beyond that, we set a trust threshold (Threshold). For the condition $C_4$, when a sensor’s trust value is smaller than a threshold, the fog node will broadcast its trust value to other sensors.
B. Least Squares Algorithm

The purpose of the least squares algorithm is to find in a closed form the value of the \( \theta \) that minimizes \( J(\theta) \). First, the cost function, \( J(\theta) \), is rewritten in matrix-vectorial notation. The training set \( X \) is the \( m \) by \( n \) matrix that contains \( m \) training samples, and each sample includes \( n \) feature inputs, denoted as \( X = [x^{(1)}T \ x^{(2)}T \ ... \ x^{(m)}T]^T \). Then, let \( \overline{y} \) be the \( m \)-dimensional vector containing all the target values from the training set: \( \overline{y} = [y^{(1)} \ y^{(2)} \ ... \ y^{(m)}]^T \). Now, because \( H_0(x^{(i)}) = (x^{(i)})^T \theta \), the algorithm can easily verify that \( X\theta - \overline{y} = [(H_0(x^{(1)}) - y^{(1)}) \ ... \ (H_0(x^{(m)}) - y^{(m)})]^T \). Thus, using the fact that for a vector \( z \), we have \( z^T z = \sum_i z_i^2 \),

\[
J(\theta) = \frac{1}{2} (X\theta - \overline{y})^T (X\theta - \overline{y}).
\]

Finally, to minimize \( J(\theta) \), we set its derivatives to zero, and obtain the normal equations \( X^T X \theta = X^T \overline{y} \). Thus, the value of \( \theta \) that minimizes \( J(\theta) \) is given in closed form by the equation: \( \theta = (X^T X)^{-1} X^T \overline{y} \). When the minimal \( \theta \) is confirmed, we have found the optimal fitting function \( H(X) \) for the given dataset.

C. Property Analysis

Theorem 1: The complexity of our trust learning approach is \( O(n^3) \), where \( n \) is the number of features.

Proof of Theorem 1: Our trust learning approach can be devided into two stages. The first stage uses the least squares algorithm to find the fitting function \( H(X) \). The time complexity of this stage is \( O(n^3) \), where \( n \) is the number of input features. The computational effort required for matrix inversion is \( O(n^3) \). Thus, in the second stage, the time complexity of our algorithm is \( O(1) \). Therefore, the worst-case time complexity of our approach is \( O(n^3) \).

Theorem 2: For a given training set, the fitting function \( H(X) \) can minimize the error of trust evaluation.

Proof of Theorem 2: The relationship hypothesis between the sensors’ features and trust value is denoted as: \( H_0(x) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n \). Then, we utilize the least squares method to compute value of \( \theta \) for \( H(\theta) \). Based on the property of the least squares algorithm, the resulting \( \theta \) can minimize the cost of the function \( J(\theta) \). Therefore, for the given training set, the fitting function \( H(X) \) can minimize the error of trust evaluation.

IV. PERFORMANCE EVALUATION

A. Parameter Set

To validate the effectiveness of our proposed trust learning approach, we conducted extensive simulations using MATLAB 2015a. The dataset contains ten samples, and each sample includes three features: the data delivery ratio, the response time \( (s) \), and the average energy consumption \( (J) \). The least squares algorithm is programmed by MATLAB and called the "regress" function. This function returns four parameters: an \( n - by - 1 \) vector of coefficient estimates, an \( n - by - 2 \) matrix of 95% confidence intervals for the coefficient estimates, and an \( n - by - 1 \) vector of residuals, and an \( n - by - 2 \) matrix of intervals, where \( n \) is the number of samples.

B. Simulation Results

Figure 2 shows the residual figure for trust learning with three features. It is obvious that there is no outlier. All the samples satisfy the hypothesis function within a 95% confidence interval. Compared with the experimental result in [7], the trust learning approach achieves a higher detection rate, approximating 1. Next, we wanted to investigate the relationship between the number of features and trust learning. Therefore, we conducted the simulation with only two features: response time and average energy consumption. As shown in Fig.3, we represent the trust learning results with two features: a three-dimensional graph and a residual figure.

In Fig.3 (a), the colorful plane represents a linear plot of the regression result, and the points represent the samples. We can see that most of samples are close to the plane, except for one outlier. The corresponding residual figure of the three-dimensional graph is shown in Fig.3 (b). Compared with Fig.2, this graph provides two useful pieces of information. On one
hand, the residuals in Fig. 3 are smaller than those in Fig. 2, but there is an outlier in Fig. 3. Therefore, we know that the approach will perform trust learning better when using more features. On the other hand, feature type selection is also a key factor. In other words, the feature selection determines the effectiveness of trust evaluation.

V. CONCLUSIONS

In this paper, we solve the trust communication problem in sensor-cloud systems. Because sensors are deployed in an unattended environment, a sensor-cloud system is vulnerable to malicious attacks, such as node capture attack, Sibyl attacks, replicated attacks, and so on. Traditional security mechanisms are not good at solving internal attacks and, thus, are not applicable to sensor-cloud systems. Beyond that, various types of trust models in the current literatures can be divided into four categories: risk factor, belief, subjective probability and transitivity relationship, but most existing trust evaluation methods are restricted to specific attacks or applications; therefore, they lack compatibility, verifiability and accuracy. To solve these problems, we formulate the trust evaluation issue as a multiple linear regression problem. Then, we adopt the least squares algorithm to find the best fitness function and verify their trust degrees. Considering the energy restriction, we introduce fog nodes to assist in trust evaluation. The experimental results show that our approach will perform trust learning better when using more features.

ACKNOWLEDGMENTS

This work was supported in part by grants from the National Natural Science Foundation (NSF) of China under Grant Nos. 61672441 and 61632009 and Operation Platform and the Foster Project for Graduate Student in Research and Innovation of Huaqiao University (No. 1511414005).

REFERENCES