A Secure IoT Service Architecture With an Efficient Balance Dynamics Based on Cloud and Edge Computing

Tian Wang, Guangxue Zhang, Anfeng Liu, Md Zakirul Alam Bhuiyan, *Senior Member, IEEE*, and Qun Jin, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT)-Cloud combines the IoT and cloud computing, which not only enhances the IoT's capability but also expands the scope of its applications. However, it exhibits significant security and efficiency problems that must be solved. Internal attacks account for a large fraction of the associated security problems, however, traditional security strategies are not capable of addressing these attacks effectively. Moreover, as repeated/similar service requirements become greater in number, the efficiency of IoT-Cloud services is seriously affected. In this paper, a novel architecture that integrates a trust evaluation mechanism and service template with a balance dynamics based on cloud and edge computing is proposed to overcome these problems. In this architecture, the edge network and the edge platform are designed in such a way as to reduce resource consumption and ensure the extensibility of trust evaluation mechanism, respectively. To improve the efficiency of IoT-Cloud services, the service parameter template is established in the cloud and the service parsing template is established in the edge platform. Moreover, the edge network can assist the edge platform in establishing service parsing templates based on the trust evaluation mechanism and meet special service requirements. The experimental results illustrate that this edge-based architecture can improve both the security and efficiency of IoT-Cloud systems.**

***Index Terms*—Edge computing, efficiency, internal attacks, Internet of Things (IoT), security, trust evaluation mechanism.**

Manuscript received June 15, 2018; revised August 5, 2018; accepted August 28, 2018. Date of publication September 13, 2018; date of current version June 19, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61872154, Grant 61772148, and Grant 61672441, in part by the Natural Science Foundation of Fujian Province of China under Grant 2018J01092, in part by the Fujian Provincial Outstanding Youth Scientific Research Personnel Training Program, in part by the Subsidized Project for Cultivating Postgraduates Innovative Ability in Scientific Research of Huaqiao University under Grant 1611314018, and in part by 2017 and 2018 Waseda University Grants for Special Research Projects under Grant 2017B-302 and Grant 2018B-288. *(Corresponding author: Qun Jin.)*

T. Wang and G. Zhang are with the College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China (e-mail: cs_tianwang@163.com).

A. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China (e-mail: afengliu@mail.csu.edu.cn).

M. Z. A. Bhuiyan is with the Department of Computer and Information Sciences, Fordham University, New York, NY 10458 USA (e-mail: zakirulalam@gmail.com).

Q. Jin is with the Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Tokorozawa 359-1192, Japan (e-mail: jin@waseda.jp).

Digital Object Identifier 10.1109/JIOT.2018.2870288

I. Introduction

THE Internet of Things (IoT) is based on a very large number of objects/things that connect to the Internet to help humans perceive the world and improve their quality of life [1]. However, there are many IoT characteristics, such as limited storage and processing capacity, that can reduce the service performance of the IoT [2]. Cloud computing can address these limitations associated with the IoT in terms of management, storage, computation, and processing. Moreover, cloud computing can create more services by integrating IoT resources. Due to these advantages, the concept of IoT-Cloud has been proposed. This concept combines the advantages of the IoT and cloud computing technologies to provide more and better services [3]. However, there are still some security and efficiency problems with IoT-Cloud that must be solved.

The IoT is vulnerable to security threats, especially internal attacks that frequently occur in the physical device layer and the network communication layer [4]. Unlike internal attacks, external attacks can be resisted by traditional security mechanisms, such as encryption, authorization, and auditing [5]. However, traditional security mechanisms cannot resist internal attacks effectively, especially in the resource-constrained IoT [6]. In an internal attack, the attacker is in possession of some IoT devices and then conducts further attacks using these captured devices [7]. The trust evaluation mechanism, which is designed to solve internal attack issues, is an effective supplement to the traditional security mechanism [8]. However, many trust evaluation mechanisms in the IoT consume a lot of resources, which has a large impact on the IoT performance and lifetime. With increasingly many types of internal attacks appearing, it is unlikely that the size of the trust evaluation mechanism will be reduced.

With the increase in the number of IoT-Cloud applications, increasingly more repeated/similar requirements are sent to the cloud. It is inefficient for an IoT-Cloud system to address these requirements one by one and constantly improving hardware performance is not a long-term solution. In addition, there are delay issues in IoT-Cloud services since the cloud is far away from users and the IoT [9]. As failed services also reduce efficiency, it is necessary to solve some uncertainties in the IoT, such as device faults, network congestion, and large environment changes. In order to avoid these situations, monitoring, and data analysis tasks can be performed, however, it is not advised to perform these tasks in the cloud.

To solve the above problems, an edge-based IoT-Cloud architecture with a trust evaluation mechanism and service template was established. Edge computing is performed at the Internet's edge with a lot of computing and storage nodes, such as gateways, routers, mobile fog nodes, and edge servers, which are close to the underlying network [10]. Edge computing also refers to cloudlets, micro datacenters, and fog nodes, which has advantages in the quick response to cloud services [11], [12]. The edge computing layer in this architecture is divided into two main parts: 1) the edge network and 2) the edge platform. The edge network is established on underlying edge nodes (move/static powerful nodes) and is parallel to the IoT. The edge platform is composed of edge nodes (edge servers) that lie between the IoT and cloud, and this platform is a central hub of the IoT-Cloud service architecture [13].

The main contributions can be summarized as follows.

1) An edge network was adopted to move a large part of the trust evaluation mechanism out of the IoT. In the IoT, devices perform the direct trust calculation and send exceptions to the edge network. The edge network collects trust information from devices and analyzes this information for the entire trust state of the IoT.
2) Service templates were established in the cloud and on the edge platform. The service parameter template in the cloud stores the matching information while the service parsing template on the edge platform stores the matching information and parsing strategies.
3) The trust evaluation mechanism was integrated into IoT-Cloud services via edge computing. In the process of IoT service strategy establishment, the trust evaluation mechanism enables the edge platform to select trusted devices in order to generate or transfer data. Moreover, the edge network can monitor the IoT network load with balance dynamics and assist the edge platform in timely adjusting strategies.

This paper is organized as follows. In Section II, related work is introduced. The basic concept of a novel architecture is presented in Section III. Section IV presents the detailed design of the edge network, the edge platform and the functions in the cloud. Experiment results and analyses are reported in Section V. The final section concludes this paper.

II. Related Work

To address internal attack issues, many trust mechanisms have been proposed. The trust mechanism plays an important role in many aspects, such as reliable data fusion and mining, user privacy protection, information security enhancement, and service assurance [14], [15].

In some cases, the IoT has both physical and social attributes. Based on this feature, Chen *et al.* [16] proposed a trust mechanism that can select effective feedback through a similarity filtering method based on friendship, social contact, and community of interest relationships. Moreover, to minimize the trust convergence time and resist recommendation trust attacks, an adaptive filtering technique is designed to find the best means of combining direct trust and recommendation trust. To resist attacks aimed at the recommendation trust, such as bad-mouthing, Alshehri *et al.* [17] proposed a scalable trust management solution based on IoT clustering to address practical and pressing issues. Of course, all storage and computation tasks are performed by physical IoT devices, which requires greater energy consumption. Considering trust derivation, Duan *et al.* [18] proposed an energy-aware trust derivation scheme that manages overheads through adopting a game theoretic approach in the trust derivation process. However, this scheme only provides partial security, and it is difficult to accurately measure the level of security.

There are many trust-based intrusion detection systems (IDSs) in wireless sensor networks (WSNs), which are used to defend against internal attacks. However, the efficiency of IDS is reduced on the IoT due to the very large amount of data that is generated during a short period. Meng *et al.* [19] proposed a Bayesian-based trust management method that incorporates traffic sampling into IDS under a hierarchical structure. Liu *et al.* [20] proposed a physical IDS-to-gateway and virtual IDS-to-gateway detection model that detects attacks against both physical sensors and virtual sensors (VSs). In IDSs, false alarm messages cannot be avoided and the intrusion detection threshold is difficult to set.

To solve data error issues in WSNs, Yang *et al.* [21] proposed a data error detection approach via which the cloud can quickly detect and locate errors in large sensor data sets. Data collection and transmission is an important part to maintain the QoS of IoT-Cloud. To ensure trustworthy data collection, Wang *et al.* [12] adopted mobile sinks to collect sensor data and used a CTDC system to evaluate the trustworthiness of both sensors and mobile sinks. To improve the data transfer environment, Zhu *et al.* [22] proposed a trust-assisted sensor-cloud (TASC) system that selects trusted sensors in WSNs and trusted data centers in the cloud to constitute the data transfer route from sensor to user. However, there are still many problems to be solved in the data trust level, such as time delays, data level attacks, and data integrity.

In IoT-Cloud, a very large number of connected devices and services emerge, causing network load issues in the centralized cloud architecture. To optimize IoT-Cloud services, Barcelo *et al.* [23] defined the service distribution problem (SDP) of IoT-Cloud as IoT-CSDP and solve this problem through linear programming. For IoT-Cloud services, one single service cannot meet users' uncertain requirements. Automatic service compositing is designed and used to realize automatic matching of services and requirements with the goal of fulfilling users' requirements with the least number of IoT-Cloud services. In a multiple Cloud-based IoT application context, Baker *et al.* [24] proposed an optimization algorithm to realize energy-aware service composition.

Service optimization problems (SOPs)—problems with service selection and service resource scheduling—are caused by the increasing number of services and increasing variety of requirements. Based on service domain features, Xu *et al.* [25] proposed a paradigm of service domain-oriented optimization algorithms with artificial bee colony algorithms to solve SOPs effectively. For services in IoT-Cloud systems, there are two issues that deserve more attention to achieve the goal of green and sustainable development; one is that many users require the same data from the cloud and the other is that multiple users request data from the cloud simultaneously. To reduce the resource cost due to these two issues, Zhu *et al.* [26] proposed an MMDD scheme. In the

process of IoT-Cloud systems providing service, many applications require data from different regions, i.e., a single virtual machine (VM) in a particular data center needs to integrate many VSs from different data centers. To ensure high QoS and maintain user satisfaction, Chatterjee *et al.* [27] designed an optimal decision rule for selecting the appropriate data center, which stores a single VM to serve users.

However, existing schemes have some shortcomings in terms of security management and service provision, such as one-sidedness, low expandability, high resource consumption, large delay, and inefficiency. Moreover, few studies consider the repeated/similar service issues encountered in many fields, such as smart transportation, healthcare, and augmented reality [28], [29]. Edge computing may present a more advantageous platform for solving these issues via an integrated approach; edge computing is an affordable and sustainable computing paradigm to provide many services [30], [31]. For the design, the edge network can reduce the unnecessary communication in the trust evaluation mechanism, maintain the IoT load balance, perform special IoT services, and ensure data transfer reliability. Moreover, the edge network is more suitable for mobile IoT. More fine-grained or integrated service templates are saved in the edge platform, which is flexible and scalable to an increasing number of IoT services. Moreover, many trust evaluation mechanisms can be established in the edge platform, such as the device fault detection, data error detection, and internal attack detection at the data level because edge computing has a lower latency and can identify problems with a smaller computational cost.

III. Preliminary

A. Trust Evaluation Mechanism

The concept of trust is based on human social relationships. There is no precise definition for such a thing in IoT-Cloud. Zhu *et al.* [22] proposed a definition in the context of wireless communications.

The trust of a node A in a node B is the subjective expectation of node A receiving positive outcomes from interaction with node B in a specific context.

B. Novel Service Architecture

The service architecture of the IoT-Cloud system is divided into three layers: 1) the data collection layer; 2) the data processing layer; and 3) the application service layer, as shown in Fig. 1. In this novel service architecture, the edge network lies in the data collection layer, the edge platform is located in the data processing layer and the application service layer, and the cloud is a part of the application service layer.

The edge network has the following primary advantages.

1) Replacing the recommendation/indirect trust.
2) Balancing the IoT load dynamically and selecting trusted devices to perform the service by establishing the entire trust state of the IoT.
3) Executing special user requirements, such as delay, integrity, and precision.

The edge platform consists of four primary functions.

1) Virtualizing physical devices into virtual devices.

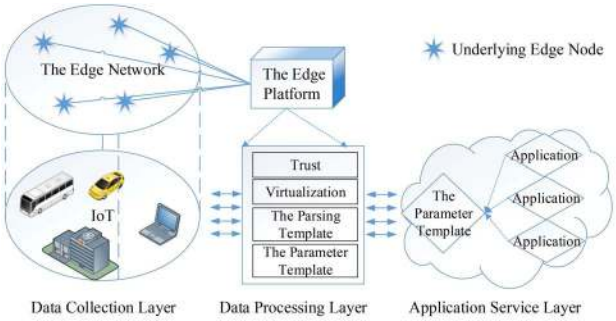


Fig. 1. Typical example for IoT-Cloud service.

2) Creating the parameter template $\text{Template}_{\text{parameter}}$ and the parsing template $\text{Template}_{\text{parsing}}$ in the edge platform.
3) Balancing dynamically the cloud load by providing some service on the edge platform.
4) Cooperating with the edge network to ensure IoT reliability at the data level.

The cloud mainly performs the following functions.

1) Parsing user requirements and finding the related $\text{Template}_{\text{parameter}}$ in the cloud, before sending the special digital information to the target edge platforms.
2) Cooperating with the edge platform to create a new $\text{Template}_{\text{parsing}}$ when there is no corresponding $\text{Template}_{\text{parsing}}$ and storing this new $\text{Template}_{\text{parameter}}$.
3) Preferentially processing services that have more stringent demands.
4) Storing a lot of historical data to be used for deeper data mining and analysis.

C. Three Basic Service Scenes

There are three basic service scenes in which the user requirements are limited to one IoT, as shown in Fig. 2. Many mixed-service scenes where the user requirements are based on multiple IoTs can be expressed as combinations of basic scenes, for example smart environment applications, such as smart transportation, healthcare [32], and augmented reality. These smart environment applications have many users and receive a large number of repeated/similar requirements.

In Scene 1, a user requires a service from IoT-1, as shown in Fig. 2(a). The user enters their requirement in an application/Web, and then the application/Web sends this requirement to Edge-1. Edge-1 digitizes this requirement and checks whether there is a corresponding $\text{Template}_{\text{parameter}}$. If there is a $\text{Template}_{\text{parameter}}$, the user requirement is completed by the corresponding $\text{Template}_{\text{parsing}}$ in Edge-1. If not, Edge-1 sends the digitized requirement to the cloud. The cloud then cooperates with Edge-1 to establish a new $\text{Template}_{\text{parsing}}$ according the specifications of the IoT, such as its functions, limitations, and precision. Finally, the cloud sends $\text{Template}_{\text{parameter}}$ to Edge-1.

In Scene 2, the required IoT-2 is closer to IoT-1. If Edge-1 has a $\text{Template}_{\text{parameter}}$ for the user's requirement, the requirement is completed by Edge-1. Edge-1 sends the special digital information to Edge-2 and receives service results from Edge-2. If Edge-1 has no $\text{Template}_{\text{parameter}}$, there are several further steps to satisfy the user's requirement, as shown in

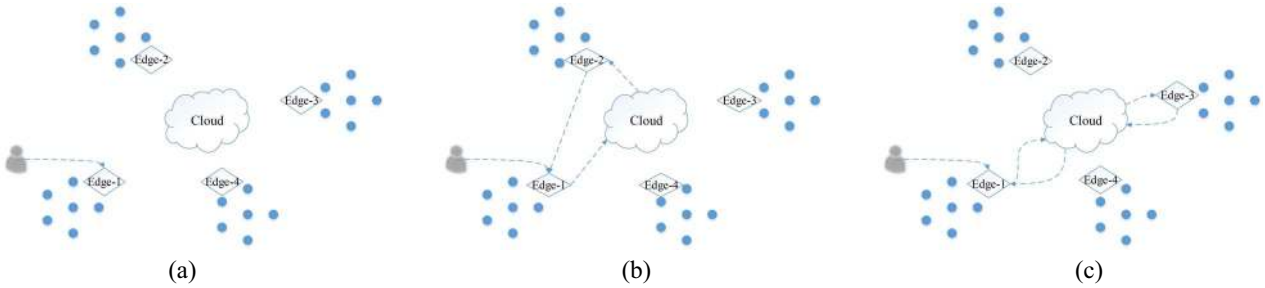


Fig. 2. Three types of service scenes. (a) Service Scene 1. (b) Service Scene 2. (c) Service Scene 3.

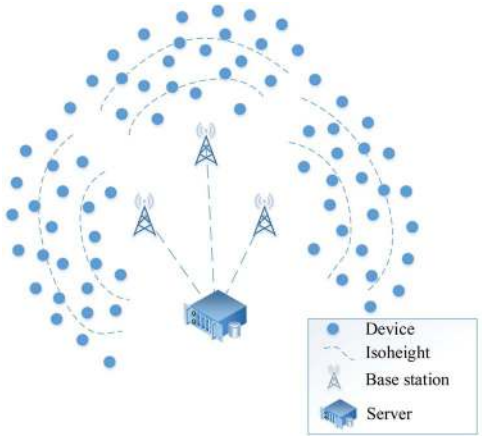


Fig. 3. Typical example of IoT.

Fig. 2(b). In step 1, the user sends the service requirement to Edge-1. In step 2, Edge-1 digitizes this requirement. In step 3, Edge-1 sends the digitized requirement to the cloud because this requirement is beyond the capability of Edge-1. In step 4, the cloud seeks $\text{Template}_{\text{parameter}}$, sends the special digital information to Edge-2, and informs Edge-2 to communicate with Edge-1. If there is no $\text{Template}_{\text{parameter}}$, the cloud cooperates with Edge-2 to establish a new $\text{Template}_{\text{parsing}}$. Finally, Edge-2 communicates with Edge-1 to send the service results to the user. Edges periodically obtain the latest $\text{Template}_{\text{parameter}}$ from the cloud, or the updating is triggered by the users' service requirements.

In Scene 3, the situation is similar to Scene 2 except that Edge-3 is far away from Edge-1. As there is no $\text{Template}_{\text{parameter}}$ in Edge-1, the requirement is completed by performing many steps, as shown in Fig. 2(c). Starting with step 4, the following steps have some differences from those described in Scene 2. In step 4, the cloud seeks a $\text{Template}_{\text{parameter}}$ and sends the special digital information to Edge-3. If there is no $\text{Template}_{\text{parameter}}$, the cloud cooperates with Edge-3 to establish a new $\text{Template}_{\text{parsing}}$. In step 5, Edge-3 executes $\text{Template}_{\text{parsing}}$ and returns the result to the cloud. Finally, the cloud returns the service result to Edge-1.

IV. Design of the Proposed Novel Service Architecture

A. Coverage of the Edge Network

The design of IoT in IoT-Cloud systems is usually hierarchical, as shown in Fig. 3. Moreover, mobile IoTs are also based on similar hierarchical architectures. It is ideal that the edge network obtains high effective coverage through fewer edge nodes in the IoT [33]. There are two means of managing edge nodes: 1) the fixed mode and 2) the moving mode.

For the fixed mode, edge nodes are placed in fixed positions of isoheight. However, this mode requires a large number of edge nodes to achieve high coverage.

For the moving mode, edge nodes have a fixed isoheight moving range; this mode needs less edge nodes but has slightly weaker real-time performance. Internal attacks do not occur all the time, so the moving mode is a good choice in certain situations. The key issue associated with this mode is determining how to dispatch edge nodes to move and how to update the trust state of the IoT in the edge network. As an example, there are three isoheights and many edge nodes, such as $isoheight-1$: (D_1, D_2, and D_3), $isoheight-2$: (D_4, D_5, D_6, D_7, D_8, and D_9), and $isoheight-3$: (D_{10}, D_{11}, D_{12}, D_{13}, D_{14}, D_{15}, D_{16}, D_{17}, D_{18}, D_{19}, D_{20}, and D_{21}). Every inner edge node communicates with two outer edge nodes, such as D_1 with (D_4, D_5), D_2 with (D_6, D_7), and D_4 with (D_{10}, D_{11}). For information interaction in the edge network, the outer edge nodes perform interactions prior to the inner edge nodes. In these circumstances, the inner edge nodes have more comprehensive knowledge regarding the IoT. The information updating is periodic or triggered by great changes, such as devices moving across areas. The general process of information updating is from the outside to the inside, and the inner node is used as the information relay point between two further nodes. If there are abnormal trust states for some devices, inner edge nodes interact with corresponding outer edge nodes to verify and even isolate malicious devices.

B. Physical Device Virtualization in the Edge Platform

The virtualization of physical devices is designed to solve the problem that one physical device cannot be shared by multiple applications. For virtualization, it is better to prepare a corresponding virtual device in advance for every physical device; these virtual devices are stored in the edge platform as finer-grained resources. As an example of data sharing, when service parsing templates need to call the physical device, the edge platform allocates corresponding virtual device interfaces to these service parsing templates, as shown in (1). The virtual device sends data to service templates and changes the parameter settings of physical devices according to relevant requirements. The key to virtualization is standardization of the data output format because the data format of physical devices may be heterogeneous, as shown in (2). A virtual

TABLE I
LIST OF ABBREVIATIONS

Abbreviations	Words	Abbreviations	Words
T	Trust	Con	Consumption
Ev	Evidence	Digi	Digitization
Td	Threshold	Req	Requirement

device receives heterogeneous data and transforms it into a standard data format. This data is then sent to service templates. Standardization of the data output format is beneficial for further data processing, such as aggregation, integration, and extraction

$$\text{Device}_{\text{physical}} \overset{\text{virtualization}}{\Rightarrow} \text{Device}_{\text{virtual}} \overset{\text{interface}}{\Rightarrow} \begin{cases} \text{Service}_1 \\ \vdots \\ \text{Service}_n \end{cases} \tag{1}$$

$$\begin{cases} \text{Format}_{\text{device}-1} \\ \vdots \\ \text{Format}_{\text{device}-\text{n}} \end{cases} \overset{\text{access}}{\Rightarrow} \begin{cases} \text{Device}^1_{\text{virtual}} \\ \vdots \\ \text{Device}^n_{\text{virtual}} \end{cases} \overset{\text{transformation}}{\Rightarrow} \text{Format}_{\text{standard}}. \tag{2}$$

C. Trust Evaluation Mechanism

Table I shows the correspondence relationship between words and abbreviations.

1) Trust Evaluation Mechanism in the IoT: In the IoT, the main body of trust evaluation mechanism is the direct trust T_{direct}, which is calculated based on evidences from direct communications among adjacent devices. These evidences may include the device residual energy, the device routing failure rate, the device communication success rate, the device data correctness rate, the device signal strength, and the device forwarding delay. These evidences can be further organized into three categories: 1) the general trust evidence; 2) the network state detection evidence; and 3) the confirmation trust evidence. In every device, the general trust evidence is stored in an array $\text{Array}_{\text{evidences}}$, which is then used to calculate T_{direct}, as

$$T_{\text{direct}} = \sum_{i=1}^{n} \text{Weight}_i \times f(x_i)$$

$$f(x_i) = \begin{cases} \frac{Ev_{\text{normal}}}{Ev_{\text{total}}}, & \text{if } \left| T^{Ev}_{\text{old}} - \frac{Ev_{\text{normal}}}{Ev_{\text{total}}} \right| < Td_1 \\ w_1 \times \frac{Ev_{\text{normal}}}{Ev_{\text{total}}} + w_2 \times T^{Ev}_{\text{old}}, & \text{if } Td_1 < \left| T^{Ev}_{\text{old}} - \frac{Ev_{\text{normal}}}{Ev_{\text{total}}} \right| < Td_2 \\ 0, & \text{else} \end{cases} \tag{3}$$

where, Weight_i is the weight value of the ith piece of evidence, which is set according to the importance of the ith piece of evidence. $f(x_i)$ is the trust value of the ith piece of evidence, whose value is between 0 and 1. n is the number of needed pieces of evidence, which may vary in different IoT environment. Ev_{normal} is the normal behavior number of times for the ith device, whereas Ev_{total} is the total monitoring number of times for the ith device. Td_1 and Td_2 are two thresholds for the difference value between the old trust value and new trust state, and w_1 and w_2 are two weight values for the old trust value and new trust state, whose values are set according to the monitoring sensitivity. The smaller the values of Td_1 and Td_2 are, the higher the sensitivity is. If the value of w_1 is larger, the value of $f(x_i)$ would converge quickly to $(Ev_{\text{normal}}/Ev_{\text{total}})$ in (3).

The network state detection evidence mainly focuses on how to reduce device communication pressure and ensure network load balancing dynamically, such as monitoring the routing failure rate and the communication collision rate to dynamically adjust the data transfer route of $\text{Template}_{\text{parsing}}$. When this type of evidence reveals outliers, they are sent to the edge network as exceptions.

The confirmation trust evidence refers to seriously abnormal behaviors that directly trigger the trust state confirmation procedure in the edge network, such as larger forwarding delays and frequent communication requests. There are two types of exceptions that trigger the trust state confirmation procedure: one is that the difference value shown in (4) is greater than Td_3 (a threshold set by the manager) during the direct trust calculation, while the other is that the confirmation trust evidence triggers this procedure

$$\text{Difference} = \left| T^{\text{new}}_{\text{direct}} - T^{\text{old}}_{\text{direct}} \right|. \tag{4}$$

2) Trust Evaluation Mechanism in the Edge Network: The edge network is parallel to the IoT and does not participate in normal IoT data transfer. In the edge network, there is a table in every edge node that stores some information about the devices and trust states, which is used to ensure that the entire IoT is credible, as shown in Table II. Time in Table II is the time since the last trust update.

There are a set of rules for the execution of the trust evaluation mechanism in the edge network.

1) The edge network periodically updates and stores the trust values of every device. The trust values of every device in the edge network are affected by three aspects: the first aspect is the trust values in the IoT (when the updating period arrives, the edge network selects trust values from trusted devices and calculates the mean trust value for every device); the second aspect is the exceptions from the IoT (when the cumulative amount of these exceptions reaches *threshold*, trust updating and the trust confirmation procedure are triggered); and the third aspect is the exceptions from the edge platform (when such exceptions happen, the trust value of the corresponding device is temporarily set to 0, and the trust confirmation procedure is triggered).
2) In the trust confirmation procedure, the edge network proactively detects the target device's behavior and monitors environment information to confirm whether the target device's data is unrealistic.
3) For trust values in the IoT, when the updating period arrives, the edge node obtains trust tables from every device in its managing scope. Every trust value in these trust tables is compared with the corresponding final trust value, and trust values beyond the tolerance range of the corresponding final trust value are flagged as outliers. If there is a high fraction of outliers in one trust table, the device is considered to be malicious.

TABLE II
INFORMATION ABOUT DEVICES AND TRUST STATES

Device ID	Trust value	Difference	Network state	Confirmation	Outlier	Position	Adjacent devices	Time
Device1	98	0.05	normal	normal	No	45^o	Device3, Device4	3
Device2	86	0.1	normal	normal	No	0^o	Device3, Device5	2
Device3	73	0.2	abnormal	normal	Yes	-90^o	Device1, Device2, Device6	1

4) The physical device does not proactively obtain recommendation trust from the edge network unless there is no trusted device to transmit data or there is a new device in the IoT. If the trust value in the physical device is not very different from the final trust value in the edge network, the edge network will not change the trust value in physical devices.
5) For isolating malicious devices, the edge node informs normal devices to remove the route that has malicious devices. The method of isolating malicious devices using the edge network is efficient, accurate, and fast.

3) Trust Evaluation Mechanism in the Edge Platform: The edge platform ensures data credibility at the data level. If possible, it is more beneficial to collect every physical devices historical trust values to be used in analyses, such as analyzing the relationship between the device performance change curve and the trust value curve, the effect of environment changes on device trust value and the connection between the load and trust state for every part of the IoT. Moreover, the edge platform has the potential to expand further to address the increasing variety of internal attack types.

Some hidden data attackers may behave normally but produce incorrect data causing users to make wrong decisions. Detecting this type of attack is difficult in the IoT because the establishment of this detection mechanism in the IoT consumes more resources, such as communication, computation, and storage. However, the edge platform can use data correlations in the IoT to address hidden data attacks.

1) Device data redundancy. For obtaining more precisely integrated and stable data, many IoTs allocate more devices to monitor the same parameter in a certain area. Thus, data from different devices fluctuate within a fixed range. In the edge platform, this data can be picked out and judgments can be performed when the period of detection arrives, as

$$T_{\text{device}} = \begin{cases} 1, & \text{if } Td_{\text{lower}} < \text{Data}_{\text{device}} < Td_{\text{upper}} \\ 0, & \text{else.} \end{cases} \quad (5)$$

The upper limit, Td_{upper}, and the lower limit, Td_{lower}, can be obtained from the underlying edge node in this area. If data are out of range, the exception would trigger the detection mechanism.

2) Device data gradualness. The monitored target has directionality, which can be orderly sensed by devices in this direction. The topology of the monitoring area is perceived by underlying edge nodes and this topology is then sent to the edge platform. According to the data gradualness and topology, the edge platform can determine the ideal data model, obtain real data from devices, and contrast this data with the ideal data model. If there are many outliers, the corresponding physical devices are considered malicious.

TABLE III
JUDGMENT CRITERIA

Total time	Total resource	Selection
$Edge.Con_{time}^{total} < Cloud.Con_{time}^{total}$	Yes	edge
$Edge.Con_{time}^{total} < Cloud.Con_{time}^{total}$	No	cloud + edge
$Edge.Con_{time}^{total} > Cloud.Con_{time}^{total}$	Yes	cloud + edge
$Edge.Con_{time}^{total} > Cloud.Con_{time}^{total}$	No	cloud

3) Some impossibilities. In monitoring areas, there are some barriers that prevent devices from detecting the target although they are close to the target. Thus, if a device breaks this impossibility, it is considered malicious.

Of course, there may be many factors that cause the trust evaluation mechanism to misjudge legitimate devices as malicious devices, such as sudden changes in the environment, temporary failure of devices, and attacks against the trust evaluation mechanism. In most cases, these devices will need to be analyzed based on their previous and future data to determine whether they are malicious. If not, it is necessary to perform trust recovery of these devices via the trust confirmation procedure.

D. Service

Since the cloud connects different areas together, services in the cloud can be divided into two categories: 1) local services and 2) remote services. In local services, users' requirements can be completed locally by the edge platform, whereas in remote services, users' requirements are completed by the cloud or both the cloud and the edge platform. The selection of local or remote services is determined by the resource consumption $\text{Con}_{\text{resource}}$ and the total time consumption $\text{Con}_{\text{time}}^{\text{total}}$, as shown in (6). There are several judgment criteria and these are shown in Table III

$$\begin{aligned} \text{Con}_{\text{resource}} &= \text{Con}_{\text{resource}}^{\text{storage}} \cap \text{Con}_{\text{resource}}^{\text{processing}} \cap \text{Con}_{\text{resource}}^{\text{bandwidth}} \\ \text{Con}_{\text{time}}^{\text{total}} &= \text{Con}_{\text{time}}^{\text{transmission}} + \text{Con}_{\text{time}}^{\text{transfer}} + \text{Con}_{\text{time}}^{\text{processing}}. \end{aligned} \quad (6)$$

In the above, $\text{Con}_{\text{resource}}^{\text{storage}}$ is the required storage space. $\text{Con}_{\text{resource}}^{\text{processing}}$ is the required hardware resource that is used to process data. $\text{Con}_{\text{resource}}^{\text{bandwidth}}$ is the bandwidth required for the user service. The values of the above three factors are 1 or 0. If the value of any factor is 0, the value of $\text{Con}_{\text{resource}}$ is 0. $\text{Con}_{\text{time}}^{\text{transmission}}$ is the time required for data transmission, $\text{Con}_{\text{time}}^{\text{transfer}}$ is the required transfer time spent on routes among the user, cloud and IoT, and $\text{Con}_{\text{time}}^{\text{processing}}$ is the time required for data processing.

Algorithm 1 Service Process

Input: User's Requirement

Output: $Array_{result}^{Edge-1}$, $Array_{result}^{cloud}$, $Result_{integration}$ // Two result sets that are in the cloud and Edge-1, respectively

$Req \rightarrow Edge-1$; // User's requirement is sent to Edge-1 in these service scenes

$Req \overset{extraction}{\rightarrow} Set(area, range_{parameter}, type_{parameter}, \cdots, type_{service})$; // Edge-1 extracts important information from the user's requirement and puts it into *Set*

$Set \overset{digitization}{\rightarrow} Array_{parameter} + Result_{integration}$; //Edge-1 digitizes *Set* into a series of numbers that correspond to $Template_{parameter}$, and the integration manner of these results is stored in $Result_{integration}$

$Array_{parameter} \overset{length}{\rightarrow} Length_{array}$;

for i from 1 to $Length_{Array}$ **do** //Finding $Template_{parameter}$ for every *Parameter*

  if $Array[i] \in Template_{parameter}$ and $Con_{resource} == 1$ in Edge-1 **then** //$Template_{parameter}$ is in Edge-1 and Edge-1 can complete this part

    Edge-1 sends ID_{cloud} in $Template_{parameter}$ to corresponding Edges;

    Edges find $Template_{parsing}$ according to ID_{cloud} and execute it;

    Edges return results to Edge-1, and Edge-1 stores these results into $Array_{result}^{Edge-1}$;

  else

    Send $Array[i]$ to the cloud;

    if $Array[i] \in Template_{parameter}$ in cloud **then** // $Template_{parameter}$ is in the cloud

      if $Cloud.Con_{time}^{total} > Edge1.Con_{time}^{total}$ and $Con_{resource} == 1$ of Edge-1 **then** //Edge-1 can independently accomplish this part with less time

        The cloud sends ID_{cloud} in $Template_{parameter}$ to corresponding target Edges;

        Edges find $Template_{parsing}$ according to ID_{cloud} and execute it;

        Edges return results to Edge-1, and Edge-1 stores these results into $Array_{result}^{Edge-1}$;

        Edge-1 updates its $Template_{parameter}$ from the cloud;

      else

        The cloud sends ID_{cloud} in $Template_{parameter}$ to corresponding target Edges;

        Edges find $Template_{parsing}$ according to ID_{cloud} and execute it;

        Edges return results to the cloud, and the cloud stores these results into $Array_{result}^{cloud}$;

      end if

    else //Creating new $Template_{parameter}$

      The cloud checks specifications of Edges and generate a new $Template_{parameter}$;

      The cloud sends parsing commands and $Template_{parameter}$ to target Edges and instructs Edges to construct a new $Template_{parsing}$;

      Edges complete services through the edge network;

      if $Cloud.Con_{time}^{total} > Edge.Con_{time}^{total}$ and $Con_{resource} == 1$ of Edge-1 **then**

        Edges return results to Edge-1, and Edge-1 stores these results into $Array_{result}^{Edge-1}$;

      else

        Edges return results to the cloud, and the cloud stores these results into $Array_{result}^{cloud}$;

      end if

    end if

  end if

end for

To free the edge platform and cloud from addressing the same or similar service requirements, it is better to create service templates ($Template_{parameter}$ and $Template_{parsing}$) in the cloud and the edge platform. $Template_{parameter}$ is created in the cloud and focuses on areas, service types, service parameters, etc. Additionally, the edge platform stores $Template_{parameter}$ to complete local service requirements. Another type of template in the edge platform is $Template_{parsing}$, which corresponds to $Template_{parameter}$. $Template_{parsing}$ pays more attention to the IoT, device, parameter setting, basic service parsing template $Template_{parsing}^{basic}$, etc. $Template_{parameter}$ consists of three parts: 1) the serial number of $Template_{parameter}$ (corresponding to the digitization of the requirement); 2) the ID_{cloud} (the special serial number that corresponds to that ID_{edge} in $Template_{parsing}$); and 3) the execution command (how the cloud sends ID_{cloud} and where the cloud sends ID_{cloud}). $Template_{parsing}$ is composed of three parts: 1) the ID_{edge}; 2) the number of strategies (one $Template_{parsing}$ may have several strategies); and 3) a series of commands (the implementation of one strategy). The service process is presented in Algorithm 1, and the result processing is rendered in Algorithm 2.

Theorem 1: All time complexities in digitization, Seeking $Template_{parameter}$ and Seeking $Template_{parsing}$ are $O(1)$.

Proof: The digitization of requirement is correlated with the method of establishing the table. As an example, assume that the number of broad categories is M, and the number of subgroupings for every broad category is a variable N.

Algorithm 2 Result Processing

Input: $Array_{result}^{Edge-1}$, $Array_{result}^{cloud}$, $Result_{integration}$
Output: Service Result

if Edge-1 satisfies $Result_{integration}$ **then** //Edge-1 can perfect final result processing
  Calculating the transmission time of $Array_{result}^{Edge-1}$ from Edge-1 to the cloud, $Edge1.Con_{time}^{transmission}$; //Method: removing the common parts of two assumptions, and compare their different parts
  Calculating the transmission time of $Array_{result}^{cloud}$ from the cloud to Edge-1, $Cloud.Con_{time}^{transmission}$;
  Getting the transfer time consumption from Edge-1 to cloud, $Con_{time}^{transfer}$;
  if $Edge1.Con_{time}^{transmission} + Con_{time}^{transfer} < Cloud.Con_{time}^{transmission}$ **then** //The expected time consumption in Edge-1 is larger than in the cloud
    Selecting the cloud as the platform for final result processing;
    Returning the final result to Edge-1 and user;
  else
    Selecting Edge-1 as the platform for final result processing;
    Returning the final result to user;
  end if
else
  Selecting the cloud as the platform for final result processing;
  Returning the final result to Edge-1 and user;
end if

The maximum number of lookups is $(M + N_{\max})$, whose time complexity is $O(1)$. Of course, there may be many levels, such as level(M, J, I). In this case, the total number of lookups is $(M + J + I + N_{\max})$, whose time complexity is also $O(1)$.

The seeking of $\text{Template}_{\text{parameter}}$ and $\text{Template}_{\text{parsing}}$ can be realized by converting from the number to the address of table, whose time complexity is $O(1)$. To reduce the size of the data table, tables are also built in a hierarchical manner, such as level(A, B, C). In this case, the number of lookups is $(A + B + C)$, whose time complexity is $O(1)$. ■

Next, the services of IoT-Cloud are introduced in detail from three aspects as follows.

1) Service in the Cloud and the Edge Platform:

$$\text{Req}_{\text{universal}} \overset{\text{extraction}}{\Rightarrow} \begin{cases} \text{area} \\ \text{type}_{\text{service}} \\ \vdots \\ \text{type}_{\text{parameter}} \end{cases} \overset{\text{digitization}}{\Rightarrow} \text{Parameter} \overset{\text{match}}{\Rightarrow} \text{Template}_{\text{parameter}} \overset{\text{distribution}}{\Rightarrow} \begin{cases} \text{Edge} - 1 \\ \vdots \\ \text{Edge} - n. \end{cases} \tag{7}$$

User requirements can be divided into two categories: 1) universal requirements $\text{Req}_{\text{universal}}$ and 2) special requirements $\text{Req}_{\text{special}}$. For $\text{Req}_{\text{universal}}$, there is a corresponding $\text{Template}_{\text{parameter}}$ in the cloud or the edge platform. The execution process of $\text{Req}_{\text{universal}}$ is shown in (7). First, some important information is extracted from $\text{Req}_{\text{universal}}$ and placed into a set Set(area, $\text{type}_{\text{service}}$, $\text{type}_{\text{parameter}}, \ldots,$ $\text{range}_{\text{parameter}}$). Second, Set is digitized as a Parameter (single service) or a set of Parameter that are Set(parameter) (combined services). Third, the cloud or the edge platform locates the corresponding $\text{Template}_{\text{parameter}}$ from its database. Finally, ID_{cloud} is sent to the corresponding edge platforms. Every $\text{Template}_{\text{parameter}}$ may have several ID_{cloud} that correspond to several edge platforms. This method not only reduces the data transfer volume but also shortens the service parsing time.

There are many notes.

1) The user requirements are from websites or Apps that have fixed formats.
2) The cloud digitizes important information as a serial number and seeks an appropriate $\text{Template}_{\text{parameter}}$ using this serial number. Of course, the digitization mostly occurs on the edge platform.
3) For the distribution, it is better to select high-trust edges. However, if there are many requirements that are in the waiting queue, the cloud should select other trusted edges unless the user has special demands.
4) If there are some new edges, the cloud should create or update new $\text{Template}_{\text{parameter}}$ according to the edges' specifications.

For $\text{Req}_{\text{special}}$, there is no $\text{Template}_{\text{parameter}}$ in the cloud or the edge platform. After digitizing important information about the users requirement as Parameter, the cloud finds appropriate edges according to Parameter and the edges' specifications. The cloud then generates $\text{Template}_{\text{parameter}}$. Subsequently, parsing commands and this new $\text{Template}_{\text{parameter}}$ are sent to the corresponding edges. Finally, the cloud guides the edge platforms to establish $\text{Template}_{\text{parsing}}$. Meanwhile, the service is finished by the edge network. If there are many edge platforms that can meet the demands, the cloud will select trusted edge platforms to complete this requirement. If edge platforms can finish it, $\text{Template}_{\text{parameter}}$ is stored in the cloud and the edge platform, and $\text{Template}_{\text{parsing}}$ is stored in the edge platform.

2) More Services in the Edge Platform: In different $\text{Template}_{\text{parsing}}$, commands tend to contain a lot of overlapping parts, so combinations of overlapping parts can fulfill these commands. These overlapping parts are designed as

$\text{Template}_{\text{parsing}}^{\text{basic}}$; these $\text{Template}_{\text{parsing}}^{\text{basic}}$ constitute more complex $\text{Template}_{\text{parsing}}$, as shown in (8). Moreover, these $\text{Template}_{\text{parsing}}^{\text{basic}}$ should be variable to address changing service requirements

$$\begin{cases} \text{Template}_{\text{parsing}}^{\text{basic}-1} \\ \quad \vdots \\ \text{Template}_{\text{parsing}}^{\text{basic}-n} \end{cases} \overset{\text{combination}}{\rightarrow} \text{Template}_{\text{parsing}}. \tag{8}$$

There are several notes.

1) Since many applications can share data from one physical device, the standard state setting of this physical device should be based on the frequency of requirements.
2) For low-frequency requirements whose state setting norms are lower than the standard state setting, the data volume can be reduced to satisfy these requirements. In this manner, the performance of the edge is guaranteed, and the service waiting time is decreased.
3) For low-frequency requirements whose state setting norms are high than the standard state settings, the edge network can be utilized to perform these services.

These $\text{Template}_{\text{parsing}}^{\text{basic}}$ and $\text{Template}_{\text{parsing}}$ are stored in the edge platform. When parsing commands from the cloud are sent to the edge platform, the edge platform directly finishes them using the edge network and creates $\text{Template}_{\text{parsing}}$. When ID_{cloud} is sent from the cloud to the edge platform, the edge platform directly seeks the corresponding $\text{Template}_{\text{parsing}}$ to finish it.

3) Service in the Edge Network: Edge computing takes on two important roles in IoT-Cloud services. The first is that edge computing can maintain the performance and QoS of the IoT. The second is that edge computing can execute special tasks without increasing the communication load of the IoT.

The edge network records the trust states of all devices in the IoT and these records are used to create data transfer routes. Moreover, in the process of creating or updating $\text{Template}_{\text{parsing}}$, the edge network provides the trust state of every device to the edge platform. The edge platform creates one or more transfer routes for every $\text{Template}_{\text{parsing}}^{\text{basic}}$ or $\text{Template}_{\text{parsing}}$. When one transfer route has some problems, $\text{Template}_{\text{parsing}}$ adopts other transfer routes to perform services. Through this method, the credibility and real-time nature of data can be well guaranteed.

Since edge computing has advantages for performing IoT functions, the edge network is employed to realize special services, such as more-real-time services, higher priority services, and more precise location services.

V. Evaluation

In this section, the performance of the trust evaluation mechanism and the service time consumption are described and discussed.

A. Parameter Settings

The simulation platform is MATLAB R2016b. The experimental setting consists of four IoTs and one cloud, where every IoT has an edge platform, and the cloud is located in the middle of the four IoTs. Edge-2 and Edge-4 are closer to Edge-1, and Edge-3 is far from Edge-1. In every IoT, there are 56 devices. These devices are randomly distributed in the IoT, and different types of devices are placed in each IoT for some specific experiments. Six underlying edge nodes are placed on two isoheights of every IoT. We set the following: for general IoT services, the transfer time from the user to the cloud is 26 ms. However, for novel IoT-Cloud services based on edge (BOE) computing, the transfer time from the user to the cloud is divided into two parts: 1) the transfer time from the user to the edge (8 ms) and 2) the transfer time from the edge to the cloud (18 ms). The transfer time between two adjacent edges is 12 ms. In the experiment, there are four types of abnormal devices: 1) temporary fault devices; 2) malicious behavior devices; 3) general abnormal devices; and 4) malicious data devices. These parameters are shown in Table IV.

TABLE IV
Experimental Parameters

Parameter	Value
Transfer time from user to cloud (ms)	26
Transfer time from user to edge (ms)	8
Transfer time from edge to cloud (ms)	18
Transfer time from Edge-1 to Edge-2 (ms)	12
Data transmission rate (bit/s)	2×10^6
Number of device in every IoT	56
Number of IoT	4
Maximum depth of every IoT	3
Number of cloud	1
Number of the edge platform	4
Number of the underlying edge node	6×4
Number of abnormal device type	4

TABLE V
Symbols and Their Corresponding Names

Symbol	Name
$T.Time_{edge}^{user}$	Transfer time from user to edge
$T.Time_{cloud}^{edge}$	Transfer time from edge to cloud
$T.Time_{edge}^{cloud}$	Transfer time from cloud to edge
$T.Time_{user}^{edge}$	Transfer time from edge to user
$Time_{digi}$	Digitization time
$Time_{match}$	Match time
$Time_{extraction}$	Information extraction time
$Time_{command}$	Command generating time
$Time_{transmission}$	Data transmission time

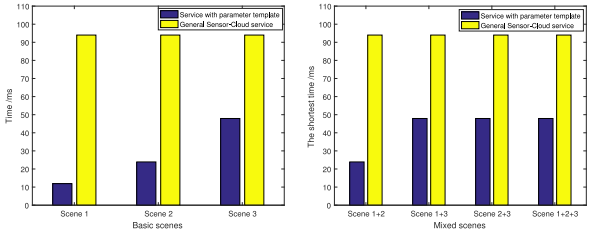
B. Service Time

In this part, the service time consists of three parts: 1) the data transfer time; 2) the data transmission time; and 3) the data processing time. For a clear comparison, the service time is divided into three phases: 1) the service time from the user to the target edge; 2) the service execution time in the IoT; and 3) the service time from the target edge to the user.

Symbols and their corresponding names are listed in Table V, and these symbols are used in the following formulas.

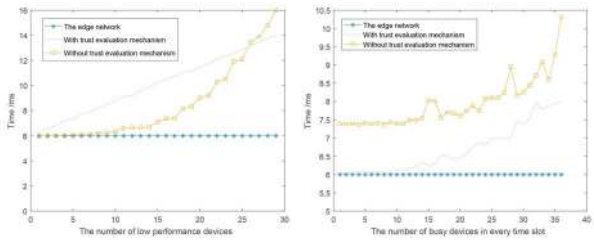
1) Service Time From the User to the Target Edge: There are some differences between service processes of the general IoT service and the novel IoT service BOE computing. These differences in the service time are demonstrated by using three basic experimental scenes and four mixed experimental scenes, as shown in Fig. 4.

Fig. 4(a) shows that the service time of "service with parameter template" is shorter than that of "general IoT-Cloud



(a) (b)

Fig. 4. Service time from the user to the target edge. (a) Three basic scenes. (b) Four mixed scenes.



(a) (b)

Fig. 5. Service execution time in IoT. (a) Influence from the number of low-performance device. (b) Influence from the degree of network congestion.

service" in the basic scenes. The reason is that service with parameter template avoids repeatedly executing the service parsing procedure.

For general IoT-Cloud service, the service time is calculated using (12). In the basic Scene 1, the service time of service with parameter template is calculated using (9), which removes $\text{Time}_{\text{command}}$, $T.\text{Time}_{\text{cloud}}^{\text{Edge}-1}$ and $T.\text{Time}_{\text{IoT}}^{\text{cloud}}$. In the basic Scene 2, the service time of service with parameter template is calculated via (10), which does not require relay via the cloud. In Scene 3, the service time of service with parameter template is calculated via (11), which has advantages in the process of parsing user requirements. $\text{Time}_{\text{match}}$ spends less time than $\text{Time}_{\text{command}}$, and $\text{Time}_{\text{digi}}$ in service with parameter template is similar to $\text{Time}_{\text{extraction}}$ of general IoT-Cloud service.

The total service time in mixed scenes is calculated through the combination of three basic equations. Service with parameter template has advantages in shortening total service time relative to general IoT-Cloud service, as shown in Fig. 4(b)

$$\text{Time}_{\text{Scene1}}^{\text{Req}} = T.\text{Time}_{\text{Edge}-1}^{\text{user}} + \text{Time}_{\text{digi}} + \text{Time}_{\text{match}} \tag{9}$$

$$\text{Time}_{\text{Scene2}}^{\text{Req}} = T.\text{Time}_{\text{Edge}-1}^{\text{user}} + \text{Time}_{\text{digi}} + \text{Time}_{\text{match}} + T.\text{Time}_{\text{Edge}-2}^{\text{Edge}-1} \tag{10}$$

$$\text{Time}_{\text{Scene3}}^{\text{Req}} = T.\text{Time}_{\text{Edge}-1}^{\text{user}} + \text{Time}_{\text{digi}} + \text{Time}_{\text{match}} + T.\text{Time}_{\text{cloud}}^{\text{Edge}-1} + T.\text{Time}_{\text{Edge}-3}^{\text{cloud}} \tag{11}$$

$$\text{Time}_{\text{general}}^{\text{Req}} = T.\text{Time}_{\text{cloud}}^{\text{user}} + \text{Time}_{\text{extraction}} + \text{Time}_{\text{command}} + T.\text{Time}_{IoT}^{\text{cloud}}. \tag{12}$$

2) Service Execution Time in the IoT: The service execution time in the IoT mainly consists of two parts: 1) the service time from edge to the target device $\text{Time}_{\text{targetdevice}}^{\text{edge}}$ and 2) the service time from the target device to edge $\text{Time}_{\text{edge}}^{\text{targetdevice}}$, as shown in (13). The service execution time based on the edge network is minimal because it can more accurately locate the target device and has shorter routes. The service execution time of "template with trust evaluation mechanism" is shorter than that of "template without trust evaluation mechanism" because the trust evaluation mechanism is more likely to select high-trust devices to transfer data and can balance the IoT load dynamically. Moreover, the trust evaluation mechanism can help $\text{Template}_{\text{parsing}}$ create several better data transfer routes to enhance the service stability. The fraction of low-performance devices and the degree of network congestion have certain influences on the service execution time, as shown in Fig. 5. Since the edge network is located outside the IoT, the fraction of low-performance devices and the degree of network congestion have weaker influences on its service execution time. Template with trust evaluation mechanism selects appropriate data transfer routes in advance; thus, it can effectively avoid low-performance devices, as shown in Fig. 5(a). As the fraction of low-performance devices exceeds 50%, the time consumption obviously increases because template with trust evaluation mechanism is more likely to select trusted devices that are in the same level. When considering the influence of network congestion on the service execution time, the number of low-performance device is set to 5. The influence of network congestion on template with trust evaluation mechanism is slightly weaker than that on "template without trust evaluation mechanism" originally, as shown in Fig. 5(b), but the phenomenon begins to reverse after the fraction of busy devices reaches 50%. The reason for this result is that template with trust evaluation mechanism tends to select trusted devices in the peer level to balance the network load

$$\text{Time}_{\text{IoT}}^{\text{Execution}} = \text{Time}_{\text{targetdevice}}^{\text{edge}} + \text{Time}_{\text{edge}}^{\text{targetdevice}}. \tag{13}$$

3) Service Time From the Target Edge to the User: To calculate the service time from the target edge to the user, the service data volume and location of the service data processing center must be considered (edge or cloud). "$1+x$" indicates that there is a very large amount of service data, and "$0+x$" indicates that there is a small amount of service data. Additionally, "$x+1$" indicates that the service data processing center lies at edge, whereas "$x+0$" indicates that the service data processing center is situated in the cloud.

For $1+x$, the service time consists of the data transmission time, the data transfer time, and the data processing time, as shown in (14). For $0+x$, the service time is composed of the data transfer and the data processing time, as shown in (15). Because the data processing time is changeable, the data transmission time and the data transfer time are the major factors considered

$$\text{Time}_{1+x}^{\text{result}} = \text{Time}_{\text{transmission}} + \text{Time}_{\text{transfer}} + \text{Time}_{\text{processing}} \tag{14}$$

$$\text{Time}_{0+x}^{\text{result}} = \text{Time}_{\text{transfer}} + \text{Time}_{\text{processing}}. \tag{15}$$

0 + 1: If users' requirements can be completed in edge, the total service time of service with parameter template would be shorter, as shown in Fig. 6. For the three basic scenes, service with parameter template can obviously shorten the service time in Scene 1 and Scene 2, as shown in Fig. 6(a), because service with parameter template avoids relay via the cloud. The service time of service with parameter template in Scene

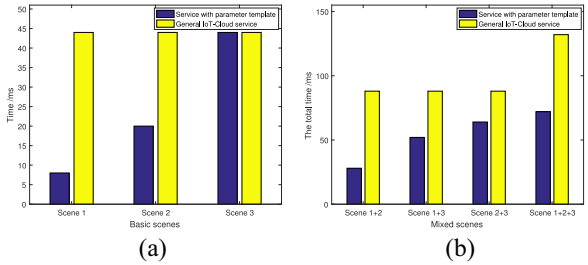


Fig. 6. Service time from the target edge to the user (0+1). (a) Basic scene. (b) Mixed scenes.

3 is almost equal to that of general IoT-Cloud service. For the four mixed scenes, the total service time of service with parameter template is shorter than that of general IoT-Cloud service because Scene 1 or Scene 2 is in the mixed scenes, as shown in Fig. 6(b).

0 + 0: In this situation, the service time of service with parameter template is almost the same as that of general IoT-Cloud service. It should not be ignored that edge has certain pretreatment capabilities and can independently perform part of mixed services. Moreover, with improving edge performance, this situation is becoming rarer.

1 + 1: When the volume of service data influences the total service time, we perform several experiments to measure the degree of this influence. This part is related to Algorithm 2. For the basic Scenes 1 and 2, it is a better choice to select Edge-1 as the data processing center, as shown in Fig. 7. If the cloud is the data processing center, there would be additional costs in the transfer time since the cloud is a relay point. For the basic Scene 3, if the data volume difference between Edge-3 and Edge-1 (Edge-3 has more data) has less time consumption $\text{Time}_{\text{transmission}}$ than $T.\text{Time}_{\text{cloud}}^{\text{Edge}-1}$, it would be a better choice that selecting Edge-1 as the data processing center. This method derives from "Remove the common parts of two assumptions, and compare their different parts." For mixed scenes, we select two mixed scenes in the experiment, as shown in Fig. 7(c) and (d). The data volume of the result in the cloud is set as $\text{Volume}_{\text{cloud}}$ (more data) and the data volume of result in the edge as $\text{Volume}_{\text{edge}}$; otherwise, Edge-1 is the better selection. Only when the data volume difference between $\text{Volume}_{\text{cloud}}$ and $\text{Volume}_{\text{edge}}$ exceeds the threshold value [the intersection points in Fig. 7(c) and (d)], the data processing center is the cloud. The threshold value represents that the time consumption of different parts whose two assumptions are the same. Mixed Scene 2 + 3 has fewer advantages than mixed Scene 1 + 3 in selecting edge as the data processing center, since the threshold in mixed Scene 2+3 is smaller than that in mixed Scene 1+3. Other mixed scenes change the threshold value through adjusting the slope of "Additional cost in the transmission time (Edge)." The higher the transmission rate is, the more likely it is that edge is selected.

1 + 0: When the service data processing center must be in the cloud, the service times of both service with parameter template and general IoT-Cloud services are almost equal. However, with enhanced edge performance, edge can address more service requirements in the future.

Note: For some mixed scenes in this novel architecture, their basic scenes can quickly perform the next service when completing part of the service.

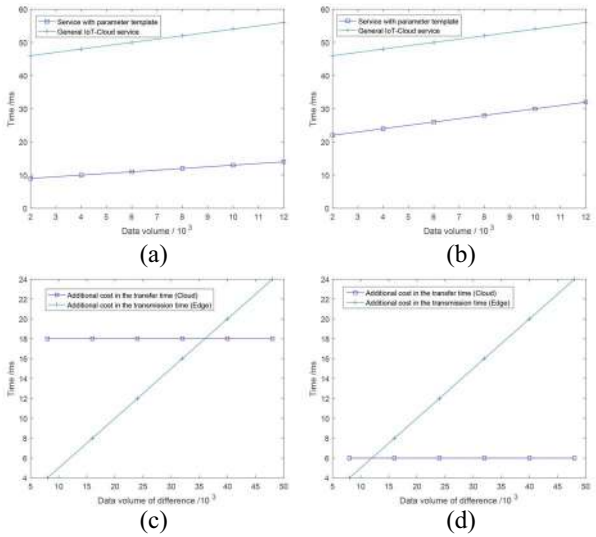


Fig. 7. Service time from the target edge to the user (1 + 1). (a) Basic Scene 1. (b) Basic Scene 2. (c) Mixed Scene 1 + 3. (d) Mixed Scene 2 + 3.

C. Trust Evaluation Mechanism Performance

In the IoT, there are four types of abnormal devices: 1) temporary fault devices; 2) malicious behavior devices; 3) general abnormal devices; and 4) malicious data devices. The change in the trust value of an abnormal device is measured after it is detected, as shown in Fig. 8. There are three types of trust evaluation mechanisms that are used for comparison: 1) BOE; 2) based on cluster head (BOCH); and 3) general combined trust (GCT, consisting of direct trust, recommendation trust or indirect trust). The difference between the BOE and BOCH trust mechanisms is that the latter belongs to the IoT and may have multihop distances to the outermost devices. In the trust evaluation mechanism, there should be periodic trust updating to ensure the latest network trust state, and this trust updating period was set to 50.

In this experiment, the "mean trust value" is calculated based on the trust values from all other devices that have interactions with the abnormal device except for the abnormal discovery device. The first discovery device performs a trust evaluation of the abnormal device and generates the "trust value of the device." The trust updating in the edge network is triggered by three types of events: 1) periodic trust updating; 2) accumulation value exception; and 3) command from the edge platform. In GCT and BOCH, the discovery device first requests recommendation values from adjacent devices or cluster heads when an abnormality is detected. The discovery device then calculates the final trust value of the abnormal device using rules, such as basing the calculation on weight.

The temporary fault device returns to normal after a short time (ten time slots); the trust value change of this device is shown in Fig. 8(a). "Trust value in the edge network" has no changes due to fewer exceptions from the IoT. However, the "mean trust value (BOE)" decreases because the discovery devices update the trust value of the abnormal device through the direct trust. In GCT, the discovery device finds and updates the trust value of the abnormal device, but the mean trust value is greater because other adjacent devices have not yet detected this abnormal device. BOCH is similar to GCT except for featuring a longer delay.

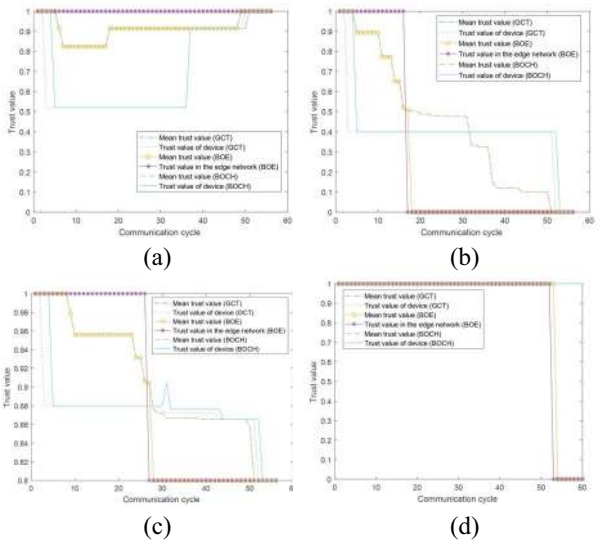


Fig. 8. Trust value changes of abnormal devices. Trust value changes of the (a) temporary fault device, (b) malicious behavior device, (c) general abnormal device, and (d) malicious data device.

When the accumulation of exceptions exceeds a threshold, the edge network updates and confirms the trust value of the abnormal device, as shown in Fig. 8(b). For GCT and BOCH, the trust value of the abnormal device is approximately 0.4 (no trust); thus, discovery devices no longer communicate with this abnormal device. When the trust updating period arrives, the abnormal device is regarded as a malicious device, and its trust value is set to 0 throughout the entire IoT.

For the general abnormal device, the execution process of BOE is similar to that in the malicious behavior detection, as shown in Fig. 8(c). The general abnormal device still has a higher trust value, so it can be selected to transfer data. When the trust updating period arrives, the final trust is calculated.

The malicious data device behaves normally but generates incorrect data to cause the user to make incorrect decisions; the trust value change of this device is shown in Fig. 8(d). This abnormality is difficult to detect unless there are some data analyses at a higher level. Because it is preferable to not perform a large amount of data analyses in the IoT, GCT, and BOCH have few advantages for the detection of malicious data devices. The edge network is located out of the IoT and can thus perform these analyses at a higher level without affecting IoT performance.

The network resource consumption is an important factor in estimating the performance of trust evaluation mechanism. Because BOE is out of the IoT, it consumes less communication resources than other schemes, as shown in Fig. 9. Moreover, BOE can perform malicious device detection at the data level, so the trust updating period can be extended to reduce the IoT resource consumption. For BOCH, its communication resource consumption is determined by the cluster depth and cluster member number (a larger depth and more cluster members require more communication resource consumption). For GCT, its communication resource consumption is associated with executing recommendations or indirect trust calculations, which is influenced by the number of adjacent devices.

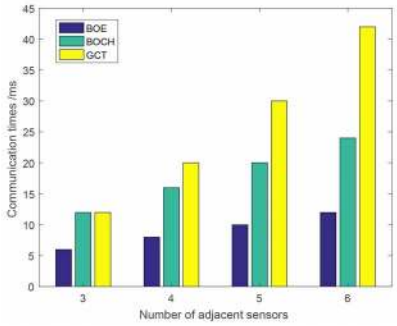


Fig. 9. IoT communication resource consumption.

VI. Conclusion

The IoT-Cloud system combines IoT and cloud computing and has gradually become a research hotspot. However, there are some security and efficiency problems that must be solved in IoT-Cloud application services. Aiming at overcoming internal attacks and repeated/similar service requirements, a novel IoT-Cloud architecture is designed based on edge computing, in which the trust evaluation mechanism ensures the IoT's security and assists the service template in solving service efficiency problems. The edge computing layer is composed of two parts: 1) the edge network and 2) the edge platform. The edge network builds on underlying edge nodes that spread over the IoT, which are responsible for ensuring the IoT's security and performing special tasks. The edge platform consists of a set of powerful edge nodes (edge servers), which perform well at parsing users' requirements through service templates with a balance dynamics. Extensive experimental results show that this novel architecture can greatly improve IoT-Cloud systems service efficiency and ensure data trustworthiness.

However, for the entire IoT-Cloud system, there is still much work to be done. Future work will seek to perfect the architecture and solve other issues through this architecture and edge computing, such as the IoT service pricing, the service scheduling, and the service access points setting.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[2] A. Botta, W. D. Donato, and V. Persico, "Integration of cloud computing and Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[3] J. Yang *et al.*, "Marine surveying and mapping system based on cloud computing and Internet of Things," *Future Gener. Comput. Syst.*, vol. 85, pp. 39–50, Aug. 2018.

[4] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: Challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 26–33, Jan. 2017.

[5] H.-W. Kim and Y.-S. Jeong, "Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain," *Human Centric Comput. Inf. Sci.*, vol. 8, no. 11, May 2018. [Online]. Available: https://doi.org/10.1186/s13673-018-0136-7

[6] A. Ahmed, K. A. Bakar, M. I. Channa, K. Haseeb, and A. W. Khan, "A trust aware routing protocol for energy constrained wireless sensor network," *Telecommun. Syst.*, vol. 61, no. 1, pp. 123–140, 2016.

[7] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 3rd Quart., 2015.

[8] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *ACM Comput. Surveys*, vol. 49, no. 1, pp. 1–35, 2016.

[9] T. Wang *et al.*, "Data collection from WSNs to the cloud based on mobile fog elements," *Future Gener. Comput. Syst.*, Jul. 2017. [Online]. Available: https://doi.org/10.1016/j.future.2017.07.031

[10] T. Wang *et al.*, "Fog-based storage technology to fight with cyber threat," *Future Gener. Comput. Syst.*, vol. 83, pp. 208–218, Jun. 2018.

[11] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[12] T. Wang *et al.*, "A novel trust mechanism based on fog computing in sensor–cloud system," *Future Gener. Comput. Syst.*, Jun. 2018. [Online]. Available: https://doi.org/10.1016/j.future.2018.05.049

[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[14] V. Suryani, S. Sulistyo, and W. Widyawan, "Internet of Things (IoT) framework for granting trust among objects," *J. Inf. Process. Syst.*, vol. 13, no. 6, pp. 1613–1627, 2017.

[15] X. Liu, Y. Liu, A. Liu, and L. T. Yang, "Defending ON–OFF attacks using light probing messages in smart sensors for industrial communication systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 3801–3811, Sep. 2018, doi: 10.1109/TII.2018.2836150.

[16] I.-R. Chen, J. Guo, and F. Bao, "Trust management for SOA-based IoT and its application to service composition," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 482–495, May/Jun. 2017.

[17] M. D. Alshehri, F. K. Hussain, and O. K. Hussain, "Clustering-driven intelligent trust management methodology for the Internet of Things (CITM-IoT)," *Mobile Netw. Appl.*, vol. 23, no. 3, pp. 419–431, 2018.

[18] J. Duan, D. Gao, D. Yang, C. H. Foh, and H.-H. Chen, "An energy-aware trust derivation scheme with game theoretic approach in wireless sensor networks for IoT applications," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 58–69, Feb. 2014.

[19] W. Meng, W. Li, C. Su, J. Zhou, and R. Lu, "Enhancing trust management for wireless intrusion detection via traffic sampling in the era of big data," *IEEE Access*, vol. 6, pp. 7234–7243, 2018.

[20] J. Liu, J. Yu, and S. Shen, "Energy-efficient two-layer cooperative defense scheme to secure sensor-clouds," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 408–420, Feb. 2018.

[21] C. Yang, C. Liu, X. Zhang, S. Nepal, and J. Chen, "A time efficient approach for detecting errors in big sensor data on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 329–339, Feb. 2015.

[22] C. Zhu *et al.*, "Trust assistance in sensor-cloud," in *Proc. Comput. Commun. Workshops*, 2015, pp. 342–347.

[23] M. Barcelo *et al.*, "IoT-cloud service optimization in next generation smart environments," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4077–4090, Dec. 2016.

[24] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, and R. Buyya, "An energy-aware service composition algorithm for multiple cloud-based IoT applications," *J. Netw. Comput. Appl.*, vol. 89, pp. 96–108, Jul. 2017.

[25] X. Xu *et al.*, "S-ABC: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition," *Future Gener. Comput. Syst.*, vol. 68, pp. 304–319, Mar. 2017.

[26] C. Zhu, V. C. M. Leung, K. Wang, L. T. Yang, and Y. Zhang, "Multi-method data delivery for green sensor-cloud," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 176–182, May 2017.

[27] S. Chatterjee, S. Misra, and S. Khan, "Optimal data center scheduling for quality of service management in sensor-cloud," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2015.2487973.

[28] T. Wang *et al.*, "Big data reduction for a smart city's critical infrastructural health monitoring," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 128–133, Mar. 2018.

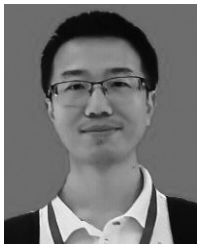
[29] M. Z. A. Bhuiyan *et al.*, "Dependable structural health monitoring using wireless sensor networks," *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 4, pp. 363–376, Jul./Aug. 2017.

[30] B. Cheng *et al.*, "Fogflow: Easy programming of IoT services over cloud and edges for smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 696–707, Apr. 2018.

[31] T. Wang *et al.*, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 3–12, Feb. 2018.

[32] Q. Jin, B. Wu, S. Nishimura, and A. Ogihara, "Ubi-liven: A human-centric safe and secure framework of ubiquitous living environments for the elderly," in *Proc. Int. Conf. Adv. Cloud Big Data*, 2017, pp. 304–309.

[33] Z. Li, Y. Liu, A. Liu, S. Wang, and H. Liu, "Minimizing convergecast time and energy consumption in green Internet of Things," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2018.2844282.



Tian Wang received the B.S. and M.S. degrees in computer science from Central South University, Changsha, China, in 2004 and 2007, respectively, and the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2011.

He is currently a Professor with Huaqiao University, Quanzhou, China. His current research interests include wireless sensor networks, fog computing, and mobile computing.

Guangxue Zhang received the B.S. degree from the Liaoning University of Technology, Jinzhou, China, in 2016. He is currently pursuing the master's degree at Huaqiao University, Quanzhou, China.

His current research interests include wireless sensor networks, mobile computing, and fog computing.

Anfeng Liu received the M.S. and Ph.D. degrees in computer science from Central South University, Changsha, China, in 2002 and 2005, respectively.

He is a Professor with the School of Information Science and Engineering, Central South University. His current research interests include cyber-physical systems, service networks, and wireless sensor network.

Dr. Liu is a member of the China Computer Federation.

Md Zakirul Alam Bhuiyan (M'09–SM'17) received the B.S. degree in computer science and technology from the International Islamic University at Chittagong, Chittagong, Bangladesh, in 2005, and the M.E. and Ph.D. degrees in computer science and technology from Central South University, Changsha, China, in 2009 and 2013, respectively.

He is currently an Assistant Professor (research) with the Department of Computer and Information Sciences, Fordham University, New York, NY, USA. He was a Post-Doctoral Fellow with Central South University, a Research Assistant with Hong Kong Polytechnic University, Hong Kong, and a Software Engineer in industry. His current research interests include dependable cyber-physical systems, wireless sensor network applications, network security, and sensor-cloud computing.

Dr. Bhuiyan has served as a Managing Guest Editor, the Program Chair, the Workshop Chair, the Publicity Chair, a TPC member, and a Reviewer of international journals/conferences. He is a member of the ACM and Center for Networked Computing.

Qun Jin (M'95–SM'17) is a Professor with the Networked Information Systems Laboratory, Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Tokorozawa, Japan. He has been extensively engaged in research works in the fields of computer science, information systems, and social and human informatics. He seeks to exploit the rich interdependence between theory and practice in his research with interdisciplinary and integrated approaches. His current research interests include human-centric ubiquitous computing, behavior and cognitive informatics, big data, data quality assurance and sustainable use, personal analytics and individual modeling, intelligence computing, blockchain, cyber security, cyber-enabled applications in healthcare, and computing for well-being.

Dr. Jin is a Senior Member of the Association of Computing Machinery and Information Processing Society of Japan.