This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing

Jun-Song Fu, Yun Liu, Fellow, IET, Han-Chieh Chao, Senior Member, IEEE, Bharat K. Bhargava, Fellow, IEEE, Zhen-Jiang Zhang, Member, IEEE

***Abstract*— With the fast development of Industrial Internet of Things (IIoT), a large amount of data is being generated continuously by different sources. Storing all the raw data in the IIoT devices locally is unwise considering that the end devices' energy and storage spaces are strictly limited. In addition, the devices are unreliable and vulnerable to many threats because the networks may be deployed in remote and unattended areas. In this paper, we discuss the emerging challenges in the aspects of data processing, secure data storage, efficient data retrieval and dynamic data collection in IIoT. Then, we design a flexible and economical framework to solve the problems above by integrating the fog computing and cloud computing. Based on the time latency requirements, the collected data are processed and stored by the edge server or the cloud server. Specifically, all the raw data are first preprocessed by the edge server and then the time-sensitive data (e.g., control information) are used and stored locally. The non-time-sensitive data (e.g., monitored data) are transmitted to the cloud server to support data retrieval and mining in the future. A series of experiments and simulation are conducted to evaluate the performance of our scheme. The results illustrate that the proposed framework can greatly improve the efficiency and security of data storage and retrieval in IIoT.**

Index Terms: Industrial internet of things, secure data storage and retrieval, fog computing, cloud computing

I. Introduction

As we step into the Internet of Things (IoT) era, terabytes of data with different sources and structures are being produced worldwide per day. In recent years, IoT has been widely used in the industrial field [1], [2], [3], [4], [5] and hence Industrial IoT (IIoT) appears. The generated data of IIoT are of great value and they can be used to run the networks or extract knowledge and rules. How to process, store and manage the data securely and efficiently is a great challenge. Fortunately, fog computing and cloud computing provide us an opportunity to solve these problems properly. Fog is close to the networks, i.e., the sources of the data, and it can access the data in a time-efficient manner. Consequently, the time-limited data should be processed and stored locally to run the network normally [6], [7]. However, storing all the data in the edge servers is unwise considering the low stability and reliability. Moreover, retrieving and mining the data stored by numerous edge servers in a distributed manner is impractical. Cloud computing is treated as a promising IT infrastructure, which can gather and organize huge IT resources to support on-demand access service in a flexible and economical manner [8]. Pushed by the data storage requirement of IIoT and attracted by these excellent features of cloud computing, an intuitive approach is outsourcing the non-time-sensitive data to the cloud [9], [10], [11], [12], [13], [14] while guaranteeing both the security and searchability of the data. Note that, though quite a large portion of the data is stored in the cloud, the whole system needs to employ the edge server as a fundamental tool. In fact, cloud computing and edge computing are interdependent with each other and they together form a service continuum between the cloud and the end devices of IIoT [15], [16], [17].

In this paper, we design a data processing framework for IIoT by integrating the functions of data preprocessing, storage and retrieval based on both the fog computing and cloud computing. The overall data processing system of IIoT consists of five main entities as shown in Fig. 1: *IIoT*, *Edge server*, *Proxy server*, *Cloud server* and *Data users*. The black arrows in the left half figure represent the process of data collection, processing and outsourcing. The red arrows mainly in the right half figure represent the process of secure data query. The IIoT continuously collects data from physical environments and then sends the data to the edge server. The time-sensitive data are first extracted and processed by the edge server and then the data will be dropped if they will not be used in the future. However, some archived data need to be preprocessed and uploaded to the cloud server for storage and retrieval. The proxy server is responsible for improving the quality of the data generated by a set networks and making the data suitable for being stored in the cloud server. Moreover, the data need to be encrypted by the proxy server while maintaining both the security and searchability which will be discussed in Section IV.C. When an authorized data user wants to obtain some specific historical data, he

JS. Fu, Y. Liu and ZJ. Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, China. The corresponding author is Y. Liu. (E-mail: {14111005, liuyun, zhangzhenjiang}@bjtu.edu.cn).

H.-C. Chao is with the School of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China; also with the School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430023, China; also with the Department of Electrical Engineering, National Dong Hwa University, Hualien 974, Taiwan; and also with the Department of Computer Science and Information Engineering, National Ilan University, I-Lan 26041, Taiwan (e-mail: hcc@mail.ndhu.edu.tw).

Bharat K. Bhargava is with the Department of Computer Science, Purdue University, IN 47906, USA. (E-mail: bbshail@purdue.edu).

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

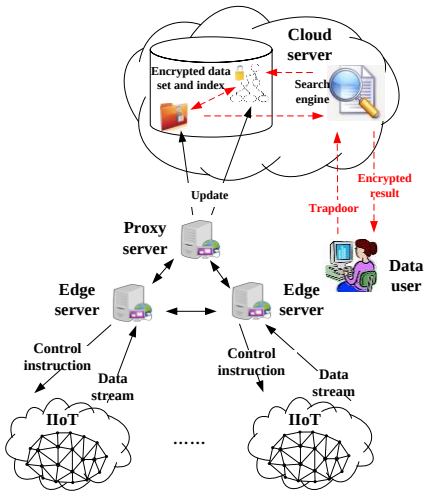

Fig. 1. The system of data process, storage and retrieval in IIoT

just needs to build a trapdoor with the help of the proxy server and then send the trapdoor to the cloud server. Based on the trapdoor, the cloud server searches the encrypted index structures by a search engine to get the result and sends the encrypted data to the data user. At last, the data user decrypts the search result to get the plaintext data.

To store the data in a secure, searchable and dynamic manner, there are many challenges and considerations in the process of designing our framework and they are presented as follows:

1. In general, the end devices of IIoT are redundantly deployed and they may collect redundant, heterogeneous, dynamic, one-sided and inaccurate data [18]. As a consequence, the raw data are not suitable for being stored and retrieved.
2. The ultimate goal of storing the data in the cloud is reusing them in the future and hence searching a specific set of data efficiently and accurately is an essential requirement for the data users.
3. For security, we need to protect the data's confidentiality without obvious decreasing of usability. Specifically, privacy-preserving schemes should be designed and employed in the data storage system.
4. Considering that the data are dynamically collected and some new data may be generated in the future, the data in the cloud need to be organized dynamically and meanwhile the index structure also needs to support dynamic update.

In this paper, we discuss the emerging challenges and basic solutions to move the IIoT forward. Specifically, we design a data processing framework for IIoT and attempt to solve the above problems properly. The rest of this paper is organized as follows: We first summarize the related work of our scheme in Section II and present the network and system models in Section III. We discuss the considerations when refining the raw data and design an object-oriented data index structure based on Retrieval Features (RF) and a privacy-preserving data retrieval scheme based on secure kNN algorithm [19] in Section IV. Moreover, a dynamic data collection method is also provided. The security of our framework is analyzed in Section V. Performance evaluation by both experiments and simulation is given in Section VI. At last, we conclude this article in Section VII.

II. Related Work

In this section, we mainly introduce the existing schemes of data storage for IoT. Recently, with the emerging of IoT and cloud computing, many IoT data storage schemes have been designed based on cloud computing in the literatures. Jiang et al. proposed a data storage framework [12] enabling efficient storage of both structured and unstructured data. The framework combines and extends multiple existing databases such as Hadoop, NoSQL database and relational database to store and manage diverse types of IoT data. The data users can access the stored data through the interfaces provided by the cloud server. However, a disadvantage of this framework is the long latency which is an inherent property of cloud-based data storage schemes. A framework including frontend layer, middle layer and backend layer [9] is proposed to seamlessly integrate IoT data storage schemes to existing enterprise information systems. This method can be easily accepted by the data owners considering that existing information systems are mature. To store a huge amount of heterogeneous data, a hybrid approach [13] is proposed to optimize data storage and retrieval which couples the document and object-oriented strategies. Moreover, some implementation details are also discussed. Kim et al. [14] designed a polynomial-time algorithm for efficiently downloading the packages from the cloud to the IoT devices. This approach can compute the amount of power allocation based on buffer backlog and the state of communication links to improve the overall performance.

Except for cloud computing, the fog computing technology is also employed to store and share the data in IoT. To support the latency sensitive data processing and storage, an efficient data sharing scheme [6] that allows smart devices to share the data with others at the edge of the IoT is proposed. In addition, the data users can search and retrieve interested data by keywords and their secret keys. Simulation result demonstrates that the proposed scheme has the potential to be effectively used in the IoT. However, the size of the network is strictly limited in the scheme and in addition it is impractical to store a large amount of data for further processing and mining considering the efficiency and security problems. Similar to our framework proposed in this paper, some other schemes also attempt to combine the fog computing and cloud computing to improve the quality of service in terms of latency, security and flexibility. Sharma et al. [20] discussed the advantages of cloud computing and edge computing, respectively. In summary, the cloud computing can construct a shared pool of computing and storage resources and the edge computing can process the data in realtime. By combining these two techniques, the proposed framework can obtain the network-wide knowledge by exploiting the historical information stored in the cloud center and the knowledge can be used to guide the edge computing to satisfy various performance requirements of heterogeneous IoT networks. An attribute-based encryption scheme is proposed in [21] to make full use of edge servers. The collected data are first encrypted by the edge

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

server before being outsourced to the cloud server. Experimental results illustrate that the edge servers bear a large portion of the workload. However, this scheme doesn't support efficient data search and hence the functionalities are limited. Choi et al. [22] took the lessons of designing an operating system from the long history of operation systems and designed a distributed operation system specifically for the IoT, i.e., FogOS, which can manage both the cloud resources and fog resources. In addition, FogOS is also a platform of incentivizing and connecting individually owned IoT devices.

III. NETWORK AND THREAT MODELS

We assume that a large number of IIoT terminal nodes are randomly deployed in an interested area to monitor the surrounding physical environment such as the work status of machines or the gas density in a factory. Each node consist the power module, perceptive module, data processing module, communication module, et al. We further assume that each pair of sensor nodes can negotiate a common session key to securely communicate with each other. The nodes in the network can transmit the data to the edge server in a relay manner by employing proper routing algorithms. The edge server is assumed to be stronger than the common IIoT nodes in both power and computing capability. We further assume that the edge server can preprocess the raw data of IIoT efficiently and execute complicated instructions to run the network properly. The edge server is connected to the proxy server and cloud server by wire or wireless links. The edge server and proxy server are assumed to be honest to the IIoT. This is reasonable considering that they are closer to the data sources in IIoT and they can be controlled by the network operators in general. For example, we may deploy an IIoT to monitor the status of industrial machines where an edge server is employed to run the network. Apparently, the edge server is deployed locally and it is totally controlled by the industrial factory. However, the cloud server is public and it is assumed to be "honest-but-curious" which is similar to the models in [23], [24]. Specifically, the cloud server can honestly execute the instructions and however it is curious to infer and analyze all the received data from the proxy server and data users.

IV. FRAMEWORK OF SECURE DATA STORAGE AND SEARCHING FOR IIOT

A. Data Integration and Fusion

Data integration and fusion is the most important basement of the total framework and it is briefly discussed as follows. In IIoT, the data are collected from multiple sources such as RFID, GPS devices and smart meters, and the data carriers can be messages, pictures, videos, numerical data, etc. Even for one type of the data carriers, such as the numerical data, the specific data models are various [25], including probability model, fuzzy set model, possibility model, rough set model, D-S evidence theory model, etc. Though integrating massive structured, semi-structured and unstructured data into a unified framework is a huge challenge, it is meaningful to merge the data and create a comprehensive and meaningful view for future utility [26]. Specifically, the data need to be first transformed to a unified resource description framework and then fused to eliminate the redundant data.

A novel concept called Information Object is proposed in [27] to model the data coming from several sources and transfer them to a unified structure for storage and mining. Further an event information management platform is designed to collect and analyze heterogeneous data streams. In [28], a resource description framework called HEP is proposed in which the representations of relational and XML event streams are integrated. To decrease the storage space in the cloud server and communication burdens of the network, the data need to be fused at different levels according to the requirements of the data users. An elaborate survey of data fusion techniques at numerical level is presented in [25]. Moreover, Thing Broker [29] is designed to integrate totally different IoT objects by employing abstracts to represent the objects, while maintaining simple and flexible interfaces for various applications.

The low quality of the data is another challenge and, some false and missing values often appear because the end devices are often unstable and unreliable. To guarantee the completeness of the dataset, the imperfect data should be eliminated by the outlier detection technique. In addition, the missing data values should also be modified and predicted by the edge server. A missing data prediction scheme for IoT is achieved in [30] by implementing the Least Mean Square dual prediction algorithm and the optimal step size is obtained by minimizing the mean-square derivation.

B. Object-oriented Data Organization and Retrieval

After getting the high-quality data, an effective index structure needs to be built in order to improve the search efficiency. In this paper, the data are organized around the monitored objects of the IIoT. For example, a set of smart devices may be deployed in an industrial machine to monitor its work status and thus all the generated data about the machine share a same identifier which is related to the monitored machine. This is reasonable considering that the data fragmentations are meaningless unless they are collected together to describe the machine. To identify the monitored objects, an easy method is assigning serial numbers to them and we can also add some basic description information into the identifiers, such as the categories. How to encrypt the object identifiers and search the interested data based on the encrypted identifiers will be discussed in Section IV.C, where an ID-AVL tree is built to improve the search efficiency.

Except for searching the data by identifiers, the data users may also want to search the data of the monitored objects by some features. For example, the data users may query the monitored data about the buildings around Purdue University which have been used for about 10 years. We assume that the top-k relevant buildings' data are needed. To support the feature-based data query, we can describe each monitored object O_i by an n-dimensional attribute vector $\mathcal{AV}_i$ which is another identifier of the stored data. An n-dimensional feature dictionary $\mathcal{A}$ with n important features is employed to regulate the feature vectors. For object O_i, $\mathcal{AV}_i[j]$ is the value on feature $\mathcal{A}[j]$ and the default value is set as 0. Simi-

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

larly, a query request is mapped to a query vector $\mathcal{Q}$. An open issue is how to transform the feature values to numerical values and define the relevance scores between the feature vectors and query vectors. In this paper, we assume that the value of each feature ranges from 0 to 1 and the relevance score between a query vector and a feature vector is defined as the inner product of the two vectors.

To improve the search efficiency of feature-based data query, we design a Retrieval Feature (RF) tree to organize the objects' feature vectors as hierarchical clusters. Given K n-dimensional feature vectors $\mathcal{AV}_1, \mathcal{AV}_2, \cdots \mathcal{AV}_K$ in a cluster, the RF vector of the cluster is defined as a quadruple (K, LS, SS, V_{max}), where K is the number of feature vectors in the cluster, LS is the linear sum of the K vectors, SS is the square sum of the K vectors, and V_{max} is defined as $V_{max}[j] = max(\mathcal{AV}_1[j], \mathcal{AV}_2[j], \cdots, \mathcal{AV}_K[j])$ and $\mathcal{AV}_i[j]$ is the j-th dimension of vector $\mathcal{AV}_i$. Based on a RF vector, the center and radius of the cluster can be easily calculated as discussed in [31]. Specifically, the center of the cluster can be calculated as follows:

$$c = LS/K. \tag{1}$$

The radius of the cluster is defined as follows:

$$R = \sqrt{\sum_{j=1}^{K}(\mathcal{AV}_j - c)^2/K}, \tag{2}$$

and it can be calculated based on the RF vector as follows:

$$R = \sqrt{(SS - LS^2/K)/K}. \tag{3}$$

As a consequence, the RF vector is an important summarization about the cluster.

A RF tree is presented in Fig. 2. It can be observed that the tree is height-balanced and we can easily infer this based on the construction process of the tree which will be presented in the following. The structure of the tree is mainly controlled by three parameters: branching factors B_1, B_2 and a threshold T. We call a node as a non-leaf node if it represents a macro-cluster and the node represents a micro-cluster is defined as a leaf node. Each non-leaf node NL_i contains at most B_1 child nodes and it is denoted as $[RF, RF_1, child_1, \cdots, RF_{B_1}, child_{B_1}]$, where RF is the RF vector of the whole cluster, RF_i is the RF vector of the i-th sub-cluster and $child_i$ is the pointer to i-th sub-cluster. A leaf node L_i contains at most B_2 feature vectors and it is defined as $[RF, child_1, \cdots, child_{B_2}]$, where RF is the RF vector of the cluster, $child_i$ is the pointer to the i-th feature vector in the cluster. Further, all the feature vectors in a leaf node must satisfy a threshold requirement, i.e., the radius of a micro-cluster has to be less than T.

The RF tree is constructed in an incremental manner and the process of inserting a feature vector $\mathcal{AV}_i$ into the RF tree is presented as follows:

1. *Identifying the leaf node:* Starting from the root node, $\mathcal{AV}_i$ recursively descends the RF tree by choosing the closest child node according to the Euclidean distances between $\mathcal{AV}_i$ and the centers of the clusters.
2. *Modifying the leaf node:* When $\mathcal{AV}_i$ reaches a leaf node L_i, we test whether node L_i can "absorb" $\mathcal{AV}_i$ without violating the constraints of B_2 and T. If so, $\mathcal{AV}_i$ is inserted into L_i and the RF vector of L_i is updated to reflect this. If not, we must split L_i to two new leaf nodes. Node splitting is done by choosing the farthest pair of feature vectors as seeds, and redistributing the remaining vectors based on the closest criteria. Apparently, the RF vectors of the two new leaf nodes need to be recalculated in this case.
3. *Modifying the path from the root node to the leaf node:* After inserting $\mathcal{AV}_i$ into a leaf node, we need to update the RF vectors of all the nodes on the path from the root node to the leaf node L_i. In the absence of a split, this simply involves updating RF vectors in order from the leaf node to the root based on Theorem 4.1 in [31]. A leaf split requires us to insert a new leaf node to the parent node. If the parent node has space for the new leaf, we just need to insert the new leaf node and then update the RF vector of the parent node. In general, however, we may have to split the parent node as well, and so up to the root. If the root is split, the tree height increases by one.

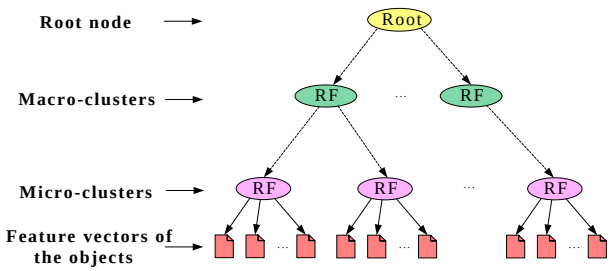

Fig. 2. The structure of a RF tree

In the RF tree, all the feature vectors of the objects are organized based on their relative similarities. It is of high probability that two similar vectors are assigned to the same cluster and this property can greatly improve the search efficiency. For a query vector $\mathcal{Q}$ provided by a data user, a parallel data search process can be easily executed in the cloud server to get the top-k relevant objects. Assume that there are l processors $\{p_1, p_2, \cdots, p_l\}$ sharing a same result list $\mathcal{RL}$ composed of the relevance scores between the query vector and the current top-k relevant objects. Given a query vector, $\mathcal{RL}$ is initialized by searching the feature vectors in the most relevant leaf node. Then, all the necessary search paths are selected from the root node to the leaf nodes based on criteria $\mathcal{Q} \cdot V_{max} > R_k$ where V_{max} is the last entry contained in a node's RF vector and R_k is the smallest relevance score in $\mathcal{RL}$. If the search can be continued on l' search paths and there are not less than l' idle processors, any l' processors are selected and each processor is responsible for searching a child path. Otherwise the redundant paths are put into a waiting queue and once an idle processor appears, it takes the oldest path in the waiting queue to continue the search. If a leaf node is scanned by a processor, the $\mathcal{RL}$ is updated for a time and apparently R_k is also updated. In the search process, quite many paths are pruned by the criteria $\mathcal{Q} \cdot V_{max} > R_k$ and the search efficiency is greatly improved. The reasonability of the criteria is given as follows. Based on the definition of V_{max}, it can be easily analyzed that the similarity between $\mathcal{Q}$ and any member in the cluster is not larger than $\mathcal{Q} \cdot V_{max}$ and hence it is also not larger than R_k. There-

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

fore, any member in the cluster cannot be a part of the search result if $\mathcal{Q} \cdot V_{max} \leq R_k$.

C. Privacy-preserving Data Search

The IIoT data are very valuable and leaking them to the cloud server or the unauthorized data users is unacceptable for the data owners. To protect the security of the data completely, the data, serial numbers and feature vectors need to be encrypted before being outsourced to the cloud server. Specifically, we encrypt them by different schemes according their different characteristics. Because the search processes are executed on the encrypted index structures and the content of the data is not used, the data can be encrypted by a proper symmetric encryption algorithm with a set of symmetric secret keys. The serial-number-based search returns the accurate results that contain a specified object serial number and an easy way is mapping the serial numbers to their hash values by a hash function. Further we design an ID-AVL tree for the hash values based on the AVL tree [32]. The AVL tree can be constructed in an incremental manner and hence it can be easily updated based on the rotation operations from time to time [32]. The most important property of the tree is that that the left child nodes of a parent node always have smaller values and the right child nodes always have larger values. Consequently, the operations of lookup, insertion and deletion a node in the tree all take $O(\log(N))$ time in both the average and worst cases where N is the number of the nodes in the tree. In the search process, the data user first hashes the serial number and then the cloud server searches the hash value in the ID-AVL tree to locate the interested data without knowing the exact serial number. In our framework, a searching process equals to searching a specific node in the tree and hence the time complexity is also $O(\log(N))$.

Another challenge is how to encrypt the RF tree while maintaining the searchability of it. In other words, we need to encrypt all the feature vectors in the tree. In this paper, an encryption scheme for the vectors is designed based on the secure kNN algorithm [19] which has been widely used in privacy-preserving document search schemes [23], [24]. The entries LS and V_{max} in a RF vector are treated as common feature vectors when encrypting the RF tree. In the encryption algorithm, the proxy server first randomly generates an $n \times n$ invertible matrix M_1. Given an feature vector $\mathcal{AV}_i$ (represented by a column vector), it is encrypted by $M_1^T \mathcal{AV}_i$ and correspondingly the trapdoor of a query vector $\mathcal{Q}$ (represented by a column vector) is built by $M_1^{-1}\mathcal{Q}$. The relevance score between $\mathcal{AV}_i$ and $\mathcal{Q}$, which is defined as the inner product of them, can be calculated by the encrypted vectors as $\mathcal{AV}_i \cdot \mathcal{Q} = (M_1^T \mathcal{AV}_i)^T M_1^{-1}\mathcal{Q}$. As a consequence, the relevance scores can be calculated accurately by the cloud server without knowing the plaintext vectors in the data search process. When a new query request is generated, the data user first maps it to a trapdoor which is then sent to the cloud server. Based on the encrypted RF tree, the feature vectors can be ranked based on the relevance scores with the trapdoor and the corresponding encrypted data are sent to the data user in order.

As proved in [19], if the cloud server can access the encrypted RF tree and trapdoors only, it cannot recover the plaintext of the RF tree. However, if the cloud server knows some other background information such as a set of plaintext feature vectors and the corresponding encrypted vectors, it may calculate M_1, and consequently the whole encrypted RF tree can be decrypted easily. To defend against this stronger threat model, we can split $\mathcal{AV}_i$ to $\mathcal{AV}_i'$ and $\mathcal{AV}_i''$, and split $\mathcal{Q}$ to $\mathcal{Q}'$ and $\mathcal{Q}''$. Specifically, a n-dimensional bit vector S is first randomly generated and if $S[j] = 0$, $\mathcal{AV}_i'[j]$, $\mathcal{AV}_i''[j]$ are set equal to $\mathcal{AV}_i[j]$, and $\mathcal{Q}'[j]$, $\mathcal{Q}''[j]$ are set to two random numbers whose sum is $\mathcal{Q}[j]$; if $S[j] = 1$, the splitting process is similar except that the roles of $\mathcal{AV}_i$ and $\mathcal{Q}$ are switched. The split vectors are encrypted by two invertible matrix M_1 and M_2, and specifically, $\mathcal{AV}_i'$ and $\mathcal{Q}'$ are encrypted as $M_1^T \mathcal{AV}_i'$ and $M_1^{-1}\mathcal{Q}'$; $\mathcal{AV}_i''$ and $\mathcal{Q}''$ are encrypted as $M_2^T \mathcal{AV}_i''$ and $M_2^{-1}\mathcal{Q}''$. In this way, the inner product of $\mathcal{AV}_i$ and $\mathcal{Q}$ can be calculated based on the four encrypted vectors [19]. The security of the scheme can be further improved by adding some artificial dimensions. Both the feature vectors and query vectors are extended from n-dimensions to $(n + n')$-dimensions and the values of $\mathcal{AV}_i[m]$, $\mathcal{Q}[m]$ $(n + 1 \leq m \leq n + n')$ are randomly generated while guaranteeing that the inner product of the added dimensions is 0. Though all the feature vectors share the same extended values, their encrypted formats are totally different by combining the split technique.

D. Dynamic Data Collection Mechanism

With the development of modern industry, more and more objects need to be monitored by the IIoT and the corresponding data need to be outsourced to the cloud. In addition, some information about the outdated objects needs to be deleted from the data set. Therefore, the ID-AVL tree and the RF tree need to be updated dynamically. When a new object appears in the IIoT, we first assign a unique identifier to the object and the hash value of the identifier is also calculated and sent to the cloud server. Then the hash value is inserted to the ID-AVL tree by the cloud server and the structure of the tree is also updated. In addition, the ID-AVL tree also supports deleting operation. Updating the RF tree is more complex and, we need to first update the plaintext RF tree in the proxy server and then synchronize the encrypted tree in the cloud server. Considering that the RF tree is incrementally constructed, it naturally supports inserting new nodes into the tree. We can also delete a feature vector from the RF tree with three steps including: *Identifying the feature vector*, *Modifying the leaf node* and *Modifying the path from the root node to the leaf node*, which is similar to the process of inserting a node into the tree. However, we may need to combine nodes rather than split nodes and the update process of RF vectors is also slightly different.

Another challenge is how to synchronize the encrypted RF tree when the unencrypted RF tree changes. Specifically, a set of update instructions should be designed to update RF vectors, split two nodes and combine two nodes. The process of generating the instructions is presented as follows:

1. *Generation of a RF vector update request:* A RF vector update request for node u in the encrypted tree is defined as $\{u, \mathrm{RF}_{new}\}$, where u is the node identifier and RF_{new} is the new RF vector of the node.

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

2. *Generation of a splitting request:* A splitting request for node u is defined as $\{u, u', \mathrm{RF}', u'', \mathrm{RF}''\}$, where u is the split node, u' and u'' are the two new nodes derived from u, and RF', RF'' are the corresponding RF vectors.
3. *Generation of a combining request:* The request of combining two nodes u', u'' is defined as $\{u', u'', u, \mathrm{RF}_{new}\}$ where u' and u'' are the two combined nodes, u is the new node, and RF_{new} is the RF vector of u.

Based on the update request, the process of updating the encrypted RF tree is presented as follows:

1. *Updating the RF vector of a node:* Once a RF vector update request $\{u, \mathrm{RF}_{new}\}$ is received by the cloud server, it replaces the original RF vector of u by RF_{new}.
2. *Splitting a node:* Once a splitting request $\{u, u', \mathrm{RF}', u'', \mathrm{RF}''\}$ is received, the cloud server first finds the parent of u and deletes the pointer to u. Then two new pointers to u' and u'' are inserted to the parent node.
3. *Combining two nodes:* Once a combining request $\{u', u'', u, \mathrm{RF}_{new}\}$ is received, the cloud server first finds the parent node of u' and u'', and then delete the pointers to u' and u''. At last, the pointer to u is inserted to the parent node.

In addition, the monitored data of the existing objects are also expanding considering that more and more data are generated by the IIoT. In the cloud, the data of an object are stored in a discrete way. For example, the proxy server can collect one object's data for a day and then outsource the data to the cloud server. Further, the data can be clustered by the features and then outsourced independently. Each minimum unit of the data is called a data point and assigned a unique identifier in the scope of a monitored object. To support fine-grained search, a function that maps the plaintext description about a data point to its identifier in the scope of a monitored object needs to be designed. In this way, the data users can retrieve the data points by a two-layer search manner, i.e., they first locate the interested objects by ID-AVL tree or RF tree, and then access the specific data points by the map function.

E. The Overall Flowchart of IIoT Data Processing and Open Issues

In this section, we present the flowchart of data processing in IIoT from a wider view as shown in Fig. 3 and then discuss some open issues. The IIoT devices are responsible for collecting data which are aggregated and fused at the data level in a collaborative way and then delivered to the edge server. After receiving the data, the edge server needs to transform them into a unified representation framework and fuse them at the feature level for convenient storage. In addition, the false data and missing data should be also processed properly. To improve search efficiency and support multiple search patterns, corresponding index structures need to be constructed and the ID-AVL tree and RF tree are built by the proxy server. To protect the feature privacy, the RF tree is encrypted by the secure kNN algorithm. At last, the proxy server outsources all the encrypted data,

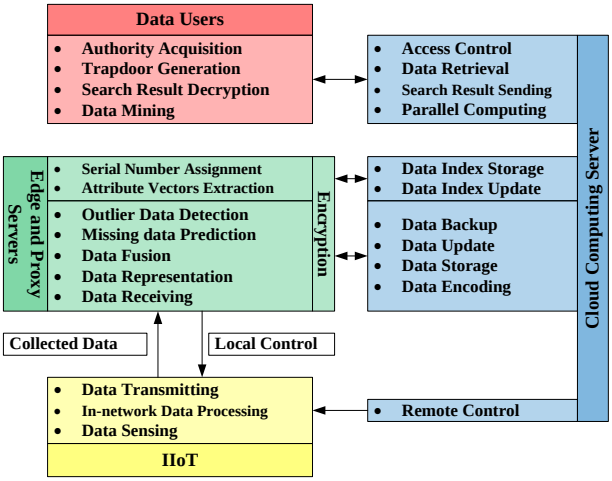

Fig. 3. The framework of data collection, storage, retrieval and mining

index structures to the cloud. When the data users want to search the data, they send the trapdoors of the queries to the cloud server which can be used to search the encrypted index structures. The cloud server is employed to store the data and execute the search operations properly. The encrypted results are sent to the data users and the data users can decrypt the encrypted data based on their secret keys.

Moreover, the data users and IIoT can communicate with the cloud server to execute specific instructions. Considering the huge amount of the IIoT data, it is likely that the data users employ the cloud server to mine the data. An important application of the cloud is parallel computing which is much more time efficient than that of traditional computing techniques. In real life, many applications are run on the cloud. In order to control the IIoT devices which may be embedded in the smart factories, the users can send instructions to the IIoT by the cloud directly. In this case, we need to employ the mature communication techniques such as 4G and WIFI.

V. Security Analysis

A. Data Security in the IIoT

In the pre-deployment phase, each IoT node is assigned with a unique identity, a public key and a secret key. Once the network is deployed, each pair of the neighboring nodes generates a shared session key to guarantee the data transmission security between the two nodes. Moreover, each IoT node needs to have a session key with the edge server. Therefore, the adversary cannot eavesdrop on the encrypted information in the data transmission process. If the adversary scatters some malicious nodes into the network to act as the common IoT nodes, they cannot negotiate the session key with any IoT node considering that they don't have the legal identity, public key and secret key. In the worst case, the adversary may compromise some IoT nodes and however he can obtain only the local data rather than all the data received by the compromised nodes. Consider an example that an IoT node sends the monitored information to the edge server and a compromised node happen to locates on the routing path. In this case, the compromised node cannot decrypt the package, because the package is encrypted by the session key of the IoT node and the edge server.

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

B. Data Security in the Edge Server and Proxy Server

As discussed in Section III, the edge server is assumed to be reliable. Once the encrypted data packages are received, the edge server first decrypts them and then processes them according to the preset instructions. At last, the time-limited data are stored in the edge server. The other data and the index structure are encrypted by the proxy server before being outsourced to the cloud server. In the process of delivering the ciphertext to the cloud server, the ciphertext is further encrypted by a symmetric encryption scheme to protect them from leaking.

C. Data Security in the Public Cloud Server

The cloud server stores a large amount of data generated by the IIoT. However, all the data are stored in ciphertext form and they cannot be decrypted without the data users' secret keys. In addition, the ID-AVL tree stores a set of hash values rather than the plaintext identifier. Meanwhile, the RF tree is encrypted by the secure kNN algorithm. Though the cloud server is curious to infer the information beneath the encrypted vectors in the encrypted RF tree, it cannot obtain the plaintext vector according to the security of the kNN algorithm [19].

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed framework based on both real experiments and simulation. A system prototype of the proposed framework is first built to monitor the temperature of a factory workshop. Then, we test our framework in terms of data transmission amount and storage space, synchronization time and data retrieval efficiency. To be fair, the proxy server is ignored in the experiments considering that it is used to guarantee the security of the framework which has been theoretically analyzed in Section V.

A. Experimental Settings

In the experiment, we deploy 6 temperature sensor nodes to form a wireless linked IIoT and the network is used to monitor the temperature of a factory workshop. The nodes communicate with each other by the ZigBee protocol. We employ a laptop with 2.6 GHz Intel Core processor, Window 7 operation system and a RAM of 4 GB to act as an edge server. As discussed previously, the edge server is responsible for running the network and preprocessing the raw data. The sensor nodes can communicate with the edge server through one-hop or multi-hop manner. A desktop computer with 3.6 GHz Intel Core processor, Windows 7 operation system and a RAM of 8 GB is employed to act as the cloud server. The edge server is connected to the cloud server through Internet and it employs the average data fusion algorithm to fuse the readings. The experimental settings are summarized in Table I.

TABLE I. EXPERIMENTAL SETTINGS

Parameter	Value
Number of nodes	6
Connection of the nodes in the IIoT	IEEE 802.15.4
Connection of the edge server and cloud server	The Internet
Edge server	Laptop (2.6 GHz Intel Core processor, Window 7 operation system and a RAM of 4 GB)
Cloud server	Desktop (3.6 GHz Intel Core processor, Windows 7 operation system and a RAM of 8 GB)
Data fusion	Average data fusion
Sensing interval	1 s

B. Data Transmission Amount and Storage Space in the Cloud Server

In this section, we assume that the sensor nodes measure the temperature every 1 second and then directly send the readings to the edge server. After collecting the data, the edge server fuses the readings and obtains an estimation of the factory workshop's temperature. The fusion results are sent to the cloud server every 10 minutes. We focus on the data transmission amount between the edge server and the cloud server, and the storage space needed to store the data in the cloud server. As presented in Fig. 4, our framework can greatly decrease both the data transmission amount and the size of storage space. This can be explained by the fact that the raw data are preprocessed before being outsourced to the cloud server. In the prototype system, 6 sensor nodes are employed and the edge server receives 6 readings per second. Therefore, without our framework, the cloud server needs to store 6 readings per second in average and in our scheme it needs to store only 1 reading per second. In theory, the data transmission amount and the storage space of our scheme is about $1/6$ to that of the original scheme. Simulation result in Fig. 4 proves the correctness of the theoretically analysis. The data transmission amount is slightly larger than that of the storage space because of the heads of the packages which contain some basic information about the packages.

C. Synchronization Time

In the proposed framework, there is an interesting trade-off between the data preprocess time and data transmission time. We employ the synchronization time to measure the time efficiency of our framework and it is defined as the

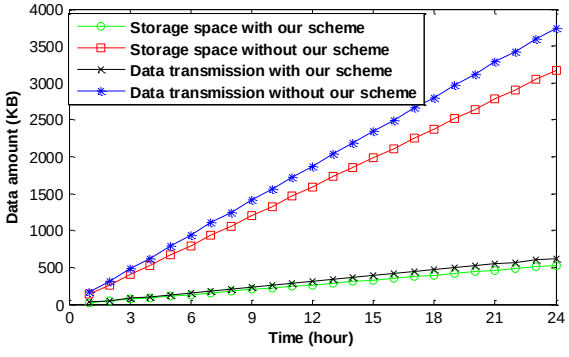

Fig. 4. Data transmission amount and storage space

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

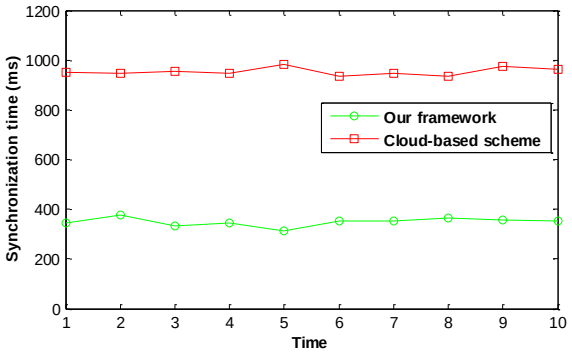

Fig. 5. Synchronization time

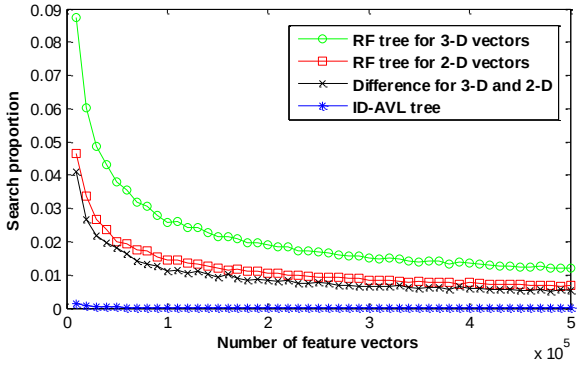

Fig. 6. Search proportion of the RF tree and ID-AVL tree

time interval between the time point that the IIoT nodes get the readings and that of the cloud server receives these readings. In this experiment, we assume that once the edge server receives all the readings of a time point, it immediately fuse the readings and send the fusion results to the cloud server. Without our framework, the edge server directly sends the received readings to the cloud server piece by piece. To be fair, the encryption process of our scheme is ignored. The total time consumption of data preprocessing and transmission is presented in Fig. 5. It can be observed that our scheme is much more time-efficient than the traditional schemes. This is reasonable considering that six links between the edge server and cloud server need to be built in the traditional schemes and in our scheme only one link is built. In addition, it can be observed that fusing the data consumes very little time compared with that of transmitting the data.

D. *Data Retrieval Efficiency*

The index structure is designed to efficiently retrieve the interested data in an extremely large database. We employ the simulation rather than experiments to thoroughly evaluate the effectiveness of the RF tree and the ID-AVL tree considering that the data generated by our sensors are too simple. We first evaluate the performance of the RF tree and ID-AVL tree in 2 and 3-dimensional feature vector spaces. The length of each feature vector is normalized to 1 and each element in a vector is a nonnegative number. As a consequence, all the vectors in 2D space locate on a quadrant. All the vectors are uniformly randomly generated. In addition, we employ the search proportion to measure the search efficiency and it is calculated by the number of the searched paths to all the paths in the tree. Each simulation is repeated for 10 times and the results are presented in Fig. 6. It can be observed that a large portion of the search paths are pruned in the process of data retrieval. In addition, the search proportion gradually decreases with the increasing of the number of feature vectors. Apparently, the search proportion of ID-AVL is much lower than that of the RF tree and it is about $O(\log(N))$ vectors need to be scanned as discussed in Section IV.C.

We further employ a real data set: the Enron Email Data Set [33] to test the search efficiency. The TF-IDF model [34], [35] is used to generate the feature vectors of the emails. We compare the proposed scheme with MRSE and present the average search time on the cloud server in Fig. 7. In MRSE, the feature vectors are stored in order and each vector needs to be scanned for a time to get the search result of a query. It can be observed that the search time in MRSE scheme [23] linearly increases with the number of feature vectors in the cloud server. This is reasonable considering that MRSE needs to scan all the feature vectors to find the search result. The RF tree greatly increases the search efficiency because the RF vectors can lead the search request to the proper feature vectors quickly. The ID-AVL tree performs the best because it is a balanced binary tree. In conclusion, our scheme performs much better than MRSE in terms of search efficiency no matter which search pattern is employed by the data users to search the interested data.

E. *Discussion*

By basically testing the prototype system, it can be observed that the proposed framework can significantly improve the data storage efficiency of existing cloud-based data storage schemes. Specifically, both the data transmission amount in the network and data storage space in the cloud server are decreased. The time consumption of data preprocessing in the edge server doesn't significantly affect the synchronization time between the edge server and cloud server. Moreover, our framework can greatly improve the search efficiency compared with scanning the database linearly. In summary, fog computing and cloud computing together can provide secure and efficient data storage for the IIoT. As a consequence, it is a trend to fuse these two promising techniques.

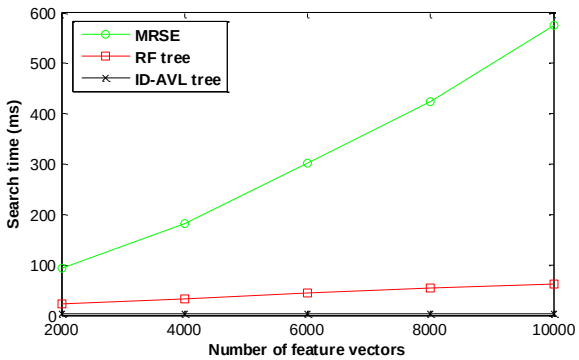

Fig. 7. Search time of the RF tree and ID-AVL tree

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

VII. Conclusions

In this paper, a secure, flexible and efficient data storage and retrieval system is designed based on both the fog computing and cloud computing techniques. The main challenges in terms of data refinement, data organization, searchable encryption and dynamic data collection are summarized, and corresponding solutions are also provided. Specifically, a retrieval features tree is designed to support efficient and accurate data retrieval and an index encryption scheme based on the secure kNN algorithm is proposed to support privacy-preserving data search. A flowchart including data mining and remote control is also presented from a wider view. The functionalities of each module are discussed in detail and some open issues are also given.

There are still many challenges in our framework. First, in this paper, only two data retrieval manners are supported and however the data users may need more flexible data search methods. Correspondingly, some new index structures need to be designed and added into the framework. Second, before mining the data, the data users need to first download the specific data sets and generalize them for the privacy sake. An interesting open research is privacy-preserving data mining in the cloud. Mature homomorphic encryption schemes are urgently needed because they can significantly improve the efficiency of our framework, as the mining process can be directly operated on the encrypted data by the cloud server.

Acknowledgement

This research is supported by Fundamental Research Funds for the Central Universities (2015YJS027) and Natural Science Foundation, No. 61772064. The corresponding author is Yun Liu.

References

[1] Xu L D. Enterprise Systems: State-of-the-Art and Future Trends[J]. IEEE Transactions on Industrial Informatics, 2011, 7(4):630-640.

[2] Li L. Technology designed to combat fakes in the global supply chain[J]. Business Horizons, 2013, 56(2):167-177.

[3] Fu J S, Liu Y, Chao H C, et al. Green alarm systems driven by emergencies in industrial wireless sensor networks[J]. IEEE Communications Magazine, 2016, 54(10):16-21.

[4] Xu L D. Introduction: Systems Science in Industrial Sectors[J]. Systems Research & Behavioral Science, 2013, 30(3):211–213.

[5] Zhang, W., Zhang, Z., & Chao, H. C. Cooperative Fog Computing for Dealing with Big Data in the Internet of Vehicles: Architecture and Hierarchical Resource Management. IEEE Communications Magazine, 2017, 55(12), 60-67.

[6] Mollah M B, Azad M A K, Vasilakos A. Secure Data Sharing and Searching at the Edge of Cloud-Assisted Internet of Things[J]. IEEE Cloud Computing, 2017, 4(1):34-42.

[7] Yi S, Hao Z, Qin Z, et al. Fog Computing: Platform and Applications[C]// Third IEEE Workshop on Hot Topics in Web Systems and Technologies. IEEE Computer Society, 2015:73-78.

[8] Ren, Kui, Cong Wang, and Qian Wang. "Security challenges for the public cloud." IEEE Internet Computing 16.1 (2012): 69-73.

[9] Shancang Li, Lida Xu, Xinheng Wang, et al. Integration of hybrid wireless networks in cloud services oriented enterprise information systems[J]. Enterprise Information Systems, 2012, 6(2):165-187.

[10] Tao F. A methodology towards virtualisation-based high performance simulation platform supporting multidisciplinary design of complex products[J]. Enterprise Information Systems, 2012, 6(3):267-290.

[11] Qing Li, Ze-yuan Wang, Wei-hua Li, et al. Applications integration in a hybrid cloud computing environment: modeling and platform[J]. Enterprise Information Systems, 2013, 7(3):237-271.

[12] Jiang L, Xu L D, Cai H, et al. An IoT-Oriented Data Storage Framework in Cloud Computing Platform[J]. IEEE Transactions on Industrial Informatics, 2014, 10(2):1443-1451.

[13] Fazio M, Celesti A, Puliafito A, et al. Big Data Storage in the Cloud for Smart Environment Monitoring ☆[J]. Procedia Computer Science, 2015, 52:500-506.

[14] Kim J. Energy-Efficient Dynamic Packet Downloading for Medical IoT Platforms[J]. IEEE Transactions on Industrial Informatics, 2015, 11(6):1653-1659.

[15] Chiang, Mung, and Tao Zhang. "Fog and IoT: An overview of research opportunities." IEEE Internet of Things Journal 3.6 (2016): 854-864.

[16] Elmroth E, Leitner P, Schulte S, et al. Connecting Fog and Cloud Computing[J]. IEEE Cloud Computing, 2017, 4(2):22-25.

[17] Masip-Bruin X, Tashakor G, Jukan A, et al. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems[J]. IEEE Wireless Communications, 2016, 23(5):120-128.

[18] Ma M, Wang P, Chu C H. Data management for internet of things: challenges, approaches and opportunities[C]//Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing. IEEE, 2013: 1144-1151.

[19] Wong, Wai Kit, et al. "Secure knn computation on encrypted databases." Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, 2009.

[20] Sharma S K, Wang X. Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks[J]. IEEE Access, 2017, 5(99):4621-4635.

[21] Huang Q, Yang Y, Wang L. Secure Data Access Control with Ciphertext Update and Computation Outsourcing in Fog Computing for Internet of Things[J]. IEEE Access, 2017, PP(99):1-1.

[22] Choi N, Kim D, Lee S J, et al. A Fog Operating System for User-Oriented IoT Services: Challenges and Research Directions[J]. IEEE Communications Magazine, 2017, 55(8):44-51.

[23] Cao, Ning, et al. "Privacy-preserving multi-keyword ranked search over encrypted cloud data." IEEE Transactions on parallel and distributed systems 25.1 (2014): 222-233.

[24] Chen C, Zhu X, Shen P, et al. An Efficient Privacy-Preserving Ranked Keyword Search Method[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(4):951-963.

[25] Khaleghi B, Khamis A, Karray F O, et al. Multisensor data fusion: A review of the state-of-the-art[J]. Information Fusion, 2013, 14(1): 28-44.

[26] Cai H, Xu B, Jiang L, et al. IoT-based big data storage systems in cloud computing: Perspectives and challenges[J]. IEEE Internet of Things Journal, 2017, 4(1): 75-87.

[27] Dao M S, Pongpaichet S, Jalali L, et al. A real-time complex event discovery platform for cyber-physical-social systems[C]//Proceedings of International Conference on Multimedia Retrieval. ACM, 2014: 201.

[28] Wang W, Guo D. Towards unified heterogeneous event processing for the Internet of Things[C]//Internet of Things (IOT), 2012 3rd International Conference on the. IEEE, 2012: 84-91.

[29] Perez de Almeida R A, Blackstock M, Lea R, et al. Thing broker: a twitter for things[C]//Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication. ACM, 2013: 1545-1554.

[30] Wu M, Tan L, Xiong N. Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications [J]. Information Sciences, 2016, 329: 800-818.

[31] Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases[C]//ACM Sigmod Record. ACM, 1996, 25(2): 103-114.

[32] Georgy Adelson-Velsky, G.; Evgenii Landis (1962). "An algorithm for the organization of information". Proceedings of the USSR Academy of Sciences (in Russian). 146: 263–266. English translation by Myron J. Ricci in Soviet Math. Doklady, 3:1259–1263, 1962

[33] W.W. Cohen, "Enron Email Data Set," https://www.cs.cmu.edu/~./enron/, 2015.

[34] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Vol. 1. No. 1. Cambridge: Cambridge university press, 2008.

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2793350, IEEE Transactions on Industrial Informatics

[35] Xia, Zhihua, et al. "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data." IEEE Transactions on Parallel and Distributed Systems 27.2 (2016): 340-352.

Jun-Song Fu received the B.E. degree from Beijing Jiaotong University, Beijing, China, in 2012. He is currently working toward the Ph.D. degree in the Key Laboratory of Communication and Information Systems, Beijing Jiaotong University. His research interests include in-network data processing, secret sharing and information privacy issues in distributed systems and Internet of Things.

Yun Liu is a Professor of Communication and Information Systems, Beijing Jiaotong University, China; Dean of Communication Engineering Department, Beijing Jiaotong University; Director of key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education; Director of Institute of Network Consensus Security, Beijing Jiaotong University. In addition, she is an evaluation expert of State Scientific and Technological reward, State Natural Sciences Fund in communication, National High Technology Research and Development Program (HTRDP). She has published more than 300 research papers and she is conducting research in wireless network security and privacy, Internet of Things and cloud computing security. She is currently a Fellow of IET, UK.

Han-Chieh Chao is a joint appointed Distinguished Professor of the Department Computer Science & Information Engineering and Electronic Engineering of National Ilan University, I-Lan, Taiwan (NIU). He is serving as the President since August 2010 for NIU as well. His research interests include High Speed Networks, Wireless Networks, IPv6 based Networks, Digital Creative Arts, e-Government and Digital Divide. He received his MS and Ph.D. degrees in Electrical Engineering from Purdue University in 1989 and 1993 respectively. He has authored or co-authored 5 books and has published about 400 refereed professional research papers. Dr. Chao was an officer of Award & Recognition for IEEE Taipei Section from 2010 to 2012 and is an IEEE senior member and a Fellow of IET (IEE)

Bharat Bhargava is a Professor of Computer Science at Purdue University. He is conducting research in security and privacy issues in distributed systems and sensor networks. This involves identity management, trust and privacy, secure routing in internet and mobile networks and dealing with malicious hosts, adaptability to attacks, controlled data dissemination, and experimental studies. Prof. Bhargava has published hundreds of research papers and won five best paper awards in addition to the technical achievement award and golden core award from IEEE. He is the editor-in-chief of three journals and serves on over ten editorial boards of international journals. Prof. Bhargava is the founder of the IEEE Symposium on Reliable and Distributed Systems, IEEE conference on Digital Library, and the ACM Conference on Information and Knowledge Management. He is a fellow of IEEE.

Zhen-Jiang Zhang received his Ph.D. degree in Communication and Information System in 2008 from Beijing Jiaotong University in 2008. He is serving as an assistant Professor since 2008 in the Department of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. Dr. Zhang has published about 60 professional research papers. His research interests are Wireless Sensors Networks techniques, including multisource data fusion, security and privacy, routing and energy management.

1551-3203 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.