Cluster Computing (2023) 26:1495–1510
https://doi.org/10.1007/s10586-022-03755-4

K-anonymity privacy-preserving algorithm for IoT applications in virtualization and edge computing

Chen Ling[1] · Weizhe Zhang[1,2] · Hui He[1]

Received: 13 January 2022 / Revised: 13 July 2022 / Accepted: 18 September 2022 / Published online: 5 October 2022

Abstract

Efficient privacy-preserving algorithms are used in internet of things (IoT) applications, virtualization and edge computing environments to decrease the high data disclosure risk. However, due to the strict security principles of current privacy-preserving algorithms, they ignore the time consumption. On the other hand, most existing privacy-preserving algorithms suffer from the overgeneralization problem, creating unnecessary information loss. Thus, two privacy-preserving algorithms are proposed in this paper for IoT applications in the virtualization and edge computing environment to address this problem and balance the information loss and disclosure risk. We first propose an overall metric to measure the performance of privacy-preserving algorithms and balance the information loss and disclosure risk. Second, the proposed algorithms can significantly decrease the time consumption by utilizing cached anonymized datasets. Moreover, the proposed algorithms also optimize the distribution of anonymized datasets and decrease the information loss by deleting the small equivalent classes. Experimental results show that our proposed algorithms are relatively fast, only consuming 34.3% operation time of current algorithms. Next, we tested the usability of anonymized datasets by putting them into an IoT application. The two datasets generate similar results, indicating that the proposed algorithms can satisfy the information loss requirements of IoT applications.

Keywords Privacy-preserving · K-anonymity · Edge computing · Data protection · IoT application

Abbreviations

IoT	Internet of things
IIoT	Industrial internet of things
Inco-Flash	Incognito-Flash
Inco-Flash2	Incognito-Flash2
PPDM	Privacy-preserving data mining
PPDP	Privacy-preserving data publishing
PUF	Physical unclonable runction
SVM	Support vector machine
PPSVCs	Privacy-preserving support vector machine classifiers
LAPP	Lightweight privacy-preserving
PDA	Privacy-preserving data aggregation scheme
DL-DOCAD	Data OffLoading and CyberAttack detection
ADULT	1994 US census database
MEPS	Medical expenditure panel survey
HC-135G	
C_{DM}	Information cost metric
J	Equivalent class set
$\|J\|$	Number of equivalent classes
$\|EG_i\|$	Equivalent class'es records number
C	Normalized information cost metric
N	Total records number
R_m	Maximum risk metric
p_{EG_i}	Probability of data disclosure of equivalent class EG_i
R_a	Average risk metric
α, β	Importance coeffecients

✉ Weizhe Zhang
wzzhang@hit.edu.cn

Chen Ling
18B903060@stu.hit.edu.cn

Hui He
hehui@hit.edu.cn

[1] School of Computer Science and Technology, Harbin Institute of Technology, Xidazhi Street, Harbin, China

[2] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China

1 Introduction

As the extension of edge computing, edge computing provides high real-time services from devices close to users. Moreover, virtualization is also used in edge computing environments to simplify the applications' deployments [1]. Due to physical constrain, existing internet of things (IoT) applications are moved to edge computing devices to improve network performance and service quality. However, most IoT applications process individuals' sensitive information, such as medical data, transaction records, and social relationship data [2]. Some researchers also found that some devices, such as industrial internet of things (IIoT), are easily attacked [3]. Moreover, Sweeney [4] found that 87% of the US people could be recognized based on other open-access datasets. He [5] also attacked the encrypted prescription dataset in South Korea, completing the de-anonymization and demonstrating the vulnerability of current encryption methods.

Above all, IoT applications in virtualization and edge computing environments need privacy-preserving algorithms to avoid individuals' sensitive information disclosure. In detail, the privacy-preserving algorithms delete or generalize part of datasets and generate anonymized datasets that satisfy security principles. Therefore, the privacy-preserving algorithms can ensure that the attackers cannot infer sensitive information based on the anonymized datasets.

1.1 Motivations and problems

Some researchers have proposed privacy-preserving algorithms to optimize anonymized datasets' information loss [6] and disclosure risk [7]. However, these privacy-preserving algorithms have some drawbacks.

The drawbacks of the current privacy-preser-ving algorithms:

Firstly, both IoT applications and edge computing environments have high real-time requirements. However, current privacy-preserving algorithms focus more on the disclosure risk [8] and ignore the time consumption. Secondly, most IoT applications require fine-grained datasets. However, with too strict security principles, current privacy-preserving algorithms cannot perfectly balance the information loss and disclosure risk. Thirdly, most privacy-preserving algorithms have the overgeneralization problem [6]. In detail, some records with unique attributes formulate small equivalent classes, and the privacy-preserving algorithms generalize whole datasets to decrease the disclosure risk of these small equivalent classes. It causes a terrible distribution of anonymized datasets and increases information loss. These drawbacks motivate us to propose a new privacy-preserving algorithm.

1.2 Contributions

This paper proposes two fast privacy-preserving algorithms, Incognito-Flash (Inco-Flash) and Incognito-Flash2 (Inco-Flash2), to minimize the time consumption and balance the disclosure risk and information loss. We develop the proposed privacy-preserving algorithms based on k-anonymity [9], a widely used privacy-preserving technology. Compared with an existing privacy-preserving algorithm, Flash [6], we propose a cache scheme in proposed algorithms to decrease time consumption. Moreover, unlike Flash, our proposed algorithms focus more on balancing information loss and disclosure risk. Finally, to address the overgeneralization problem ignored by Flash, we proposed the Inco-Flash2 to optimize the distribution of anonymized datasets. Moreover, to verify the effectiveness of the proposed algorithms, we design an IoT application architecture in edge computing environments.

The contributions of our work are summarized as follows:

- To minimize the time consumption, we propose a cache scheme for anonymized datasets in our proposed algorithm, Inco-Flash. Based on this cache scheme, Inco-Flash caches each attribute's anonymized datasets to avoid generalizing the same datasets.
- We propose an overall metric to measure anonym-ized datasets' information loss and disclosure risk. Therefore, Inco-Flash can effectively balance the information loss and disclosure risk by selecting the anonymized dataset with the minimum overall metric.
- To address the overgeneralization problem and decrease unnecessary information loss, we propose Inco-Flash2 with the recognition and deletion schemes for small equivalent classes. Therefore, Inco-Flash2 can effectively optimize the distribution of anonymized datasets and provide fine-grained datasets for IoT applications.
- To verify the usability of the anonymized datasets, we propose an IoT application architecture and apply the proposed privacy-preserving algorithms to it. In detail, we simulate the training process of a binary classification application with both anonymized and raw datasets, an example of IoT applications. In simulation experiments, the classification results generated by the anonymized and raw datasets are similar. Therefore, We ensure that the anonymized datasets are still usable for IoT applications.

The simulation experiments conducted on open-access datasets show that the time consumption of our proposed algorithms, Inco-Flash and Inco-Flash2, is only 34.3% of

an existing algorithm, Flash [6]. In addition, we analyze the distribution of anonymized datasets, showing that Incognito-Flash2 solves the overgeneralization problem. In conclusion, our proposed algorithms consume about 20 seconds for over 30,000 records, which means less than 1 millisecond per record. Therefore, based on the usability of anonymized datasets and each record's low average time consumption, the IoT applications' Quality of Service won't be affected by our proposed privacy-preserving algorithms.

The article is formed as follows. Section 2 describes related work on privacy-preserving algorithms and IoT applications. In Sect. 3, we introduce the design of IoT application architecture. In Sect. 4, we introduce the model of the privacy-preserving problem. In Sect. 5, we introduce the design of Inco-Flash and Inco-Flash2. In Sect. 6, we measure the performance of these two algorithms in terms of time consumption and distribution of the equivalent classes. The performance results are compared with the Flash. Finally, Sect. 7 summarizes our algorithms and draws some conclusions.

Table 1 Abbreviations used in this paper

Abbreviation	Meaning
IoT	Internet of things
IIoT	Industrial internet of things
Inco-Flash	Incognito-Flash algorithm
Inco-Flash2	Incognito-Flash2 algorithm
PPDM	Privacy-preserving data mining
PPDP	Privacy-preserving data publishing
PUF	Physical unclonable function
SVM	Support vector machine
PPSVCs	Privacy-preserving support vector Machine classifiers
LAPP	Lightweight privacy-preserving
PDA	Privacy-preserving data aggregation Scheme
DL-DOCAD	Data OffLoading and CyberAttack Detection
ADULT	1994 US census database
MEPS HC-135G	Medical expenditure panel survey

2 Related work

This section summarizes the research on privacy-preserving, network virtualization, and IoT applications. Table 1 shows abbreviations used in this paper.

2.1 Network virtualization researches

Researchers have proposed several resource detection, resource management, and intrusion detection method for network virtualization environments. Samira Rezaei et al. [10] summarized the classification resource management method. They found that resource management is a critical system of virtualization network environment. Guojun Wang et al. [11] proposed an SDN resource management method, cTMvSDN, to optimize the response time and SDN Quality of service. Based on Markov-Process and Time Division Multiple Access (TDMA) protocol, cTMvSDN can predict the mapping requests in next time gaps and adjust the network environment dynamically. However, they only evaluate cTMvSDN with a simulated dataset far from the real-world environment. Seyedeh Maedeh Mirmohseni et al. [12] proposed a resource allocation method based on the Markov Learning Utilization model to allocate the network resource for the request in fog computing environments. The proposed method can effectively reduce the latency and maximize network utilization by predicting the short-term resource requirement. Nevertheless, they only evaluated the proposed method in a simulated environment, different from real-world fog computing environments. Sanaz Kazemi Abharian et al. [13] proposed an intrusion detection system in cloud computing environments. The proposed system effectively extracts the attack pattern from the network state by selecting correlated features. However, this system can only detect the attack behavior included by training sets, and the attacks outside training sets won't be detected by this system.

2.2 Privacy-preserving researches

The current privacy-preserving algorithms are widely used in data analyzing and publishing processes to decrease the disclosure risk [8]. Traditionally, several researchers developed privacy-preserving algorithms based on encryption methods. Pachilakis et al. [14] focus on the shortage of certificate authorities' revocation checks of web browsers. The proposed protocol can significantly reduce the certificate authorities' revocation risk and check latency. However, they didn't use real-world datasets to evaluate their protocol, which may have a long distance to realization. Athanasios et al. [15] focus on the security in IoT networks. They implement an intrinsic physical unclonable function (PUF) on IoT devices and evaluate their security performance. Nevertheless, they didn't analyze the deployment condition and reliability of PUF. Weizhe et al. [16] designed a smartphone privacy-preserving system to detect the operation state of smartphones and prevent attacks on the smartphone sensors. This system effectively ensures the privacy of smartphones. However,

the attackers may terminate the proposed privacy-preserving system through system kernel code, increasing the data leakage risk. [17] further improves the privacy-preserving on Android. When the smartphones are missing, it locks these smartphones by the instructions from the servers. However, this privacy-preserving method can only operate on Android devices, limiting the performance of other IoT devices. Sengupta et al. [18] focus more on the path validation in the networks. They address that path validation should satisfy two privacy principles: path and index privacy. Therefore, they propose a privacy-preserving network path validation protocol named PrivNPV, satisfying these two principles. Nevertheless, they ignore the disclosure risk, and attackers may extract sensitive information from the transmitted dataset. Xueqiao et al. [19] proposed a searchable encryption scheme to encrypt the datasets with low latency in a distributed system. The proposed encryption scheme can resist keyword guessing attacks, efficiently securing the datasets. However, the proposed encryption scheme creates extra communication costs, decreasing the service quality.

Considering the limitations of encryption methods in the privacy-preservation algorithms, some researchers adopt data segmentation to protect the relationships between different datasets. [20] conducted a camouflage method to hide the relationships between data subsets, which protected the privacy of an open dataset in the data analysis. However, the proposed method cannot protect the information of heavy-tailed distributed datasets well. [21] proposed an information decomposition algorithm for set-valued data that helps with the set-valued data anonymization process. Nevertheless, they ignore the usability of anonymized datasets in data processing applications.

Moreover, some researchers utilized probability theory and statistics to design privacy-preserving algorithms. Kengpei et al. [22] proposed a privacy-preserving Support Vector Machine (SVM) classifiers (PPSVCs). The authors verified the robustness and classification accuracy of PPSVCs under attacks. However, they didn't consider the equivalent classes' distribution of PPSVCs, which may cause high information loss. Mohamed et al. [23, 24] compared the present third-party auditor models' performance. They proposed a lightweight privacy-preserving (LAPP) method in a cloud environment to reduce the auditing time and increase the confidence level of the anonymized dataset. Nevertheless, they ignore auditing accuracy and performance of each user, influencing the service fairness.

Yao et al. [25] proposed a clustering algorithm with different security requirements, Diff-BIRCH. Diff-BIRCH inserts the Laplace noise in the datasets, making the cluster results unable to be easily accessed. However, they ignored the time consumption of the proposed algorithm. Yuan and Sheng [26] designed a privacy-preserving algorithm for neural networks based on AdaBoost.M2. This algorithm operates on distributed neural networks and divides the dataset for each neural network to prevent disclosure. Based on the control of pseudo-loss, this algorithm can ensure that the anonymization won't influence the classification ability of neural networks. However, they ignored the disclosure risk that happened in data transmission. Bettahally et al. [7] proposed a privacy-preserving association rule mining method based on genetic algorithm to find the global association rules and preserve datasets' privacy. Nevertheless, the proposed algorithm may create an overgeneralization problem.

Khondker et al. [27] focused on the privacy disclosure problem in the social network datasets. They proposed a privacy-preserving method to extract the patterns of datasets and anonymize the datasets. However, they only focused on the privacy-preserving demands of part users, and too many users may consume more operation time. Xiang et al. [28] realized a privacy-preserving platform for medical data collection, transmission, and sharing, MNSSp3. MNSSp3 utilizes IoT devices to transmit and anonymize partial datasets. However, they didn't evaluate the performance of the proposed platform. On the distributed neural network LEARNAE, Spyridon and Ioannis [29] designed a distributed filesystem, IPFS, for data transmission and a decentralized network, IOTA, based on IoT devices. Based on the system they designed, they realized the privacy-preserving of LEARNAE, which also maintained the utilization of LEARNAE. Nevertheless, the attackers may recover the raw datasets based on the anonymized dataset transmitted by IOTA. Waranya Mahanan et. al. [30] propose a data privacy-preservation heuristic algorithm, extended-OIGH, on Identical Generalization Hierarchy datasets. Based on the structure of Identical Generalization Hierarchy datasets, extended-OIGH can prune part of the unnecessary anonymization process to decrease the algorithm time consumption. Nevertheless, extended-OIGH can only be applied in Identical Generalization Hierarchy datasets.

Moreover, some researchers focus on the privacy disclosure risk of database query results. Stephan Kessler et. al. [31] propose a privacy-preserving middleware in SAP High-performance ANalytic Appliance system to generalize each query results. Moreover, the proposed middleware keeps the new records that satisfy the k-anonymity principle and hides the non-anonymous ones. However, they ignored the optimization of information loss and disclosure risk. Shuai et al. [32] proposed a privacy-preserving data aggregation scheme (PDA) for cloud computing environments. The proposed scheme prices and packs the sensitive dataset, which keeps a small encrypted dataset size and a

low overhead. However, the time consumption of the proposed scheme is not evaluated in their experiments.

Moreover, Pang et al. [33] proposed a similarity-based text retrieval privacy-preserving scheme to anonymize the query results of unauthorized users. This scheme avoids attackers accessing the raw datasets based on recovering queries and documents. However, the ranking of the query results may be crippled by the proposed scheme, decreasing the usability of query results. Palanisamy Balaji et al. [34] proposed a framework to provide anonymized datasets for different users. The authors calculated the information loss and the disclosure risk in a large graph dataset. However, the proposed anonymization framework only focuses on graph datasets, unsuitable for other datasets. Jianwei Qian et. al. [35] applied the knowledge graphs to infer the sensitive information. They deanonymize over 60% of anonymized records of two real-world social network datasets. However, the knowledge graphs are generated based on generalization trees, which cannot be satisfied by some real-world datasets.

2.3 IoT deep learning applications researches

The current IoT applications use deep learning models to realize the data sets' classification, clustering, regression, etc. Most deep learning models take a long time for the model training process. And the trained deep learning models can quickly output the result with high accuracy. Di Zhu et al. [36] proposed a location classification model based on the graph convolution network. It uses the pedestrian moving trajectory and street view image to classify the places' functional characteristics. They evaluate this model in a real urban dataset, and the experiment results show that the model has high accuracy in predicting place function characteristics. This network is trained in a centralized framework, creating a high disclosure risk.

In addition, Fei Chen et al. [37] propose an IoT framework based on a deep learning model to predict and diagnose the faults of wind power generation. This framework effectively predicts the fault types to help the engineers manage the wind power generations. However, the security problem is ignored in this framework. Masoumeh Etemadi et al. [38] propose an auto-scaling mechanism based on a deep learning model to manage the fog servers' resources required by IoT devices. This mechanism optimizes network workload and network latency, increasing resource utilization. Nevertheless, datasets collected by IoT devices are not protected. Anwer Mustafa Hilal et al. [39] propose a data offloading and cyberattack detection (DL-DOCAD) technique for edge computing environments. DL-DOCAD can maximize the throughput and detect network attacks for edge computing environments. However, the computing resources requirements may exceed the IoT devices' hardware conditions.

In conclusion, these related works have some critical limitations:

- IoT applications usually have high real-time requirements. Most existing methods only focus on the data disclosure risk of anonymized datasets, ignoring the time consumption.
- The researchers develop and train IoT applications based on fine-grained datasets. The privacy-preserving algorithm should balance information loss and disclosure risk.
- Some researchers use global association rules to anonymize and recover datasets quickly. This method creates the overgeneralization problem and increases information loss.

3 Details of IoT application architecture

Before the privacy-preserving problem definition and solvation in IoT applications, we propose an IoT application architecture in edge computing environments to simulate the operation of privacy-preserving algorithms. The IoT application architecture can be divided into the cloud, edge, and end layers. The cloud layer consists of the cloud datacenter and mainly focuses on the IoT applications' developments [40]. The cloud layer periodically collects the anonymized dataset from the edge layer, and the IoT is developed based on the collected datasets. Because the IoT is developed with up-to-date datasets, it can extract the features from the present environment with high accuracy and quality.

The edge layer consists of the edge servers near the users. The IoT applications are operated in each edge server to provide the data processing service. Most IoT applications are high real-time applications, which can quickly extract features from the datasets.

The end layer consists of all the end devices, such as sensors and mobile devices. These devices collect the raw datasets from the environment and require the data processing results [41].

The data disclosure happens in the datasets transmission between end and edge layers. The attackers may hijack network connections, end devices, or even edge servers to access sensitive information. Therefore, only protecting IoT applications is not efficient in this architecture. The privacy-preserving algorithm is needed in the end layer to protect sensitive information. The detailed architecture is shown in Fig. 1.

4 Model of privacy-preserving problem

This section introduces the privacy-preserving problem model and the anonymized dataset's performance metrics. Table 2 shows the notations used in this paper.

4.1 Definition of k-anonymity principle

K-anonymity [9], the most widely used privacy-preserving principle, can effectively avoid the sensitive information disclosure of open datasets. For any k $(k > 1)$, if the anonymized dataset satisfies k-anonymity, any possible attribute combinations must have at least k number of different records. Therefore, the anonymized datasets ensure that the attackers cannot distinguish any record from more than $k - 1$ other records.

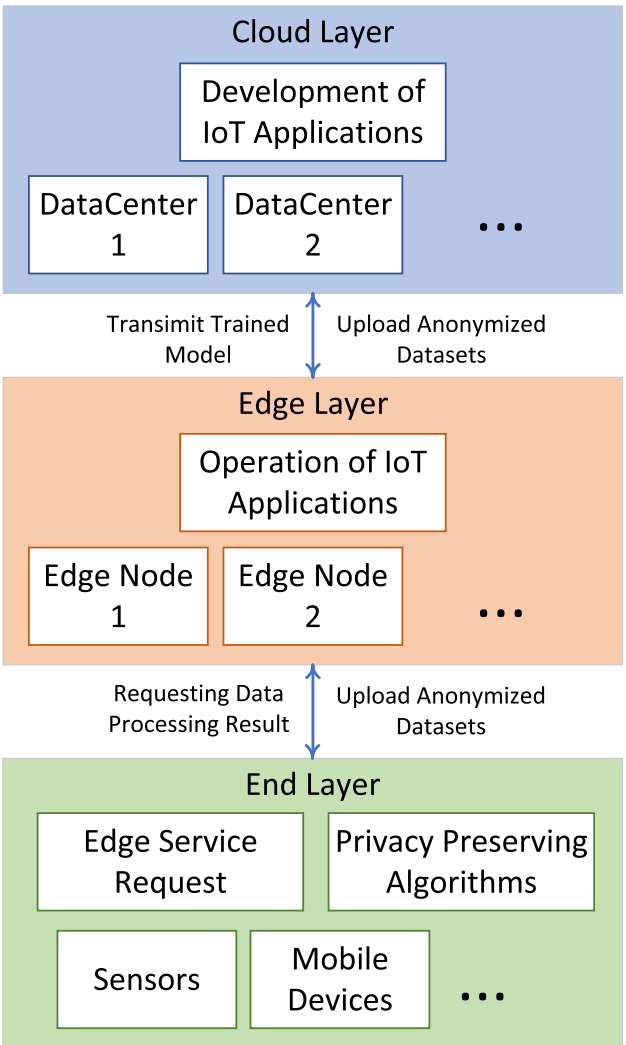

Fig. 1 System architecture of edge computing deep learning application

Table 2 Notations used in this paper

Notation	Meaning
C_{DM}	Information loss
J	Equivalent class set
$\|J\|$	Number of equivalent classes
EG	Equivalent class
$\|EG_i\|$	Records number of equivalent class
C	Normalized information loss
N	Total records number
p_{EG_i}	Probability of data disclosure
R_a	average disclosure risk
α, β	Importance parameters

The k-anonymity principle effectively decreases disclosure risk, but most dataset generalization plans are not optimal. Although generalization makes the dataset satisfy the k-anonymity principle, it also causes dataset information loss. The lower the disclosure risk, the greater the information loss. Therefore, most k-anonymity privacy-preserving algorithms aim to find the optimal generalization plan.

To formulate the problem model of privacy-preserving algorithms, we build a generalization tree to provide a basis for privacy-preserving algorithms. For example, Fig. 2 is a generalization tree for Marital-Status. Marital-Status can be divided into Married-Civ-Spouse, Divorced, Never Married, etc. These marital statuses can be generalized into two states: Spouse Present and Spouse Not Present. Finally, these two states are generalized to the $*$ value, which is the most ambiguous.

Then, we construct the solution space for the privacy-preserving problem and formulate a new search strategy. We combine the generalization plans of each attribute and

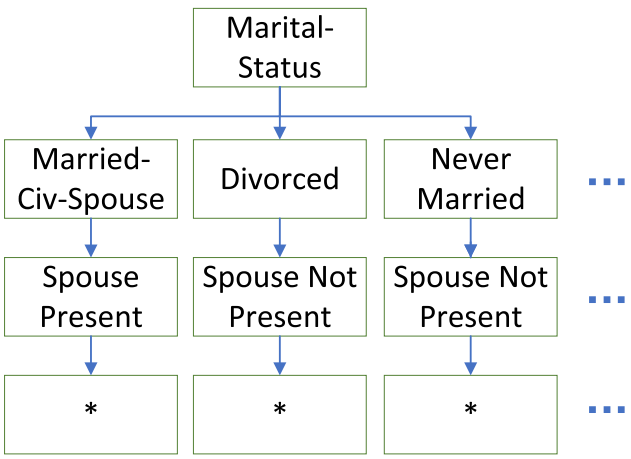

Fig. 2 Generalization tree of marital status

formulate the anonymity lattice, including all possible generalization plans. For example, we take three attributes whose maximum generalization levels are 1, 2, and 4. The detailed anonymity lattice of these three attributes is shown in Fig. 3. The numbers in nodes represent the generalization level of the attributes. Moreover, the level of a node is the sum of the generalization levels of all attributes. The bottom node (0, 0, 0) represents the raw dataset, meaning that all attributes' generalization levels are 0. In addition, the highest generalization state node (1, 2, 4) is located at the 7th level of the anonymity lattice.

The anonymity lattice has some properties that can simplify the search process. As shown in Fig. 3, if the node (1, 2, 0) does not meet the k-anonymity principle, node (1, 2, 0)'s predecessors also cannot meet the k-anonymity principle. Therefore, the red background nodes are non-anonymous, and we do not need to verify them. If it has been verified that node (1, 0, 3) is a node that meets the k-anonymity principle, all successors of (1, 0, 3) are anonymous nodes.

Privacy-preserving algorithms aim to find an optimal generalization plan in the anonymity lattice. This optimal generalization plan should meet the k-anonymity principle and keep the information loss as low as possible.

4.2 Performance measurement of anonymity scheme

While satisfying the k-anonymity principle, each anonymous node has information loss and disclosure risk. We need effective metrics to compare anonymous nodes' information loss and disclosure risk. Moreover, the optimal generalization plan should balance the information loss and disclosure risk. For the information loss metric, we use the information cost metric [42] to evaluate the information loss. The anonymized datasets have equivalent classes, including at least k number of the same records. The more equivalent classes are, the more data processing tools can extract the details in the datasets, and the lower the information loss is. Then, the detailed metric is as follows.

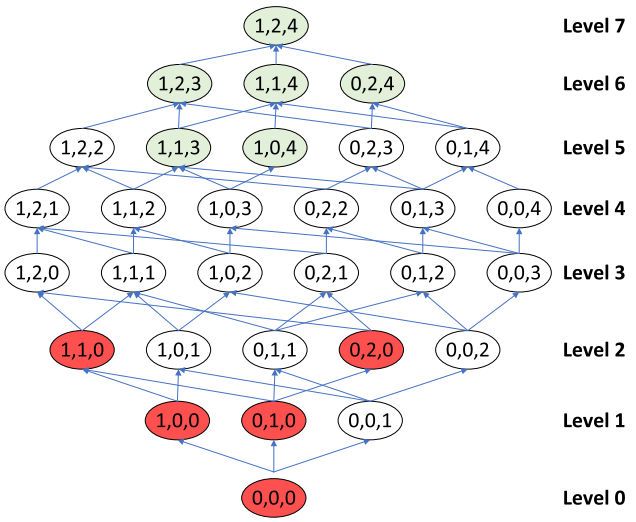

Fig. 3 Anonymity lattice

$$C_{DM} = \sum_J |EG_i|^2, \tag{1}$$

where J represents the equivalent class set, and $|EG_i|$ represents the number of records contained in the equivalent class EG_i. Moreover, we normalize the information cost metric as follows.

$$C = \frac{\sum_J |EG_i|^2}{N^2}, \tag{2}$$

where N represents the total records number of the dataset. After normalization, the value of information loss will be between 0-1.

Then, we use the average risk metric [43] to calculate the disclosure risk. The average risk metric is calculated by the weighted summation of the equivalent classes' disclosure risks. The details are as follows.

$$\begin{aligned} R_a = \frac{1}{N}\sum_J |EG_i| p_{EG_i} = \frac{1}{N}\sum_J |EG_i| \\ \times \frac{1}{|EG_i|} = \frac{1}{N}\sum_J 1 = \frac{|J|}{N}, \end{aligned} \tag{3}$$

where $|J|$ is the number of equivalent classes. p is the probability of data disclosure in the equivalent class EG_i. Finally, we propose an overall metric of the generalization plans, including information loss and disclosure risk. Here, we use the weighted summation method, which uses two variables, α and β, to control the importance of the two measurement methods and $\alpha + \beta = 1$. The details are as follows.

$$Value = \alpha \times C + \beta \times R_a. \tag{4}$$

The overall metric can effectively balance the information loss and disclosure risk in IoT applications based on α, β. For example, when $\alpha = 1, \beta = 0$, the privacy-preserving algorithm will only focus on information loss and ignore the disclosure risk. Furthermore, if $\alpha = 0, \beta = 1$, it will only focus on disclosure and ignore the other. In the following experiments, we set $\alpha = \beta = 0.5$, which means that we see the importance of information loss and disclosure risk is equal.

5 The improvement of the anonymization algorithm

Florian Kohlmayer et al. [6] proposed the privacy-preserving algorithm named Flash. The process of Flash has the following steps: First, it finds a longest path and then checks whether the node at $[\frac{1}{2}path.size - 1]$ meets the k-anonymity principle. Second, it performs a binary search to find the local optimal generalization plan on this path and adds all non-anonymous nodes into a minimum heap. Third, a new path is formed from the nodes in the minimum heap and repeats the first and second steps until all nodes are checked. Flash [6] decreases the time consumption, but it still has some shortcomings.

- Flash can only use the predecessors' anonymized datasets in the current node check process. However, most cached anonymized datasets are useless to Flash.
- Flash consumes much time to find a better generalization plan than the local optimal. It is an unnecessary waste for IoT applications with high real-time requirements.
- Small equivalent classes in anonymized datasets cause overgeneralization problems.

Therefore, we propose the privacy-preserving algorithms, Inco-Flash and Inco-Flash2, to solve these problems based on a cache scheme. We cached the anonymized datasets at each level. Therefore, the nodes in the same levels can use the cached anonymized datasets to save time consumption.

5.1 The procedure of Inco-Flash

Like Flash, Inco-Flash checks each node based on the properties of anonymity lattice, including five parts: path find, node check, data cache, path check and outer loop. The detailed procedure of each part is shown as follows.

The path find algorithm is a depth-first search from the inputted node to the top node, which constructs a path composed of unchecked nodes. It searches the longest path with unchecked nodes from the inputted starting node to the bottom node. Then, this longest path will be checked by node check and path check algorithms.

In node check and path check algorithms, the data cache algorithm stores the anonymized datasets and generate *cacheList*. In detail, for each anonymized dataset, the data cache algorithm caches the anonymous ones. For the non-anonymous datasets, the data cache algorithm will record that their nodes are non-anonymous. Finally, the path check algorithm will access the data cache record before calling the node check algorithm. If *cacheList* has the dataset that needs to check, the path check algorithm will directly use the cached anonymized datasets in the following process.

The node check algorithm is responsible for the k-anonymity principle check of each node. It generalizes each attribute and divides the raw dataset into several equivalent classes based on the node's value. Algorithm 1 shows the process of the node check algorithm.

Algorithm 1: Node Check of Inco-Flash

```
input  : node, the node needs to be checked;
         dataset, raw dataset;
         attributeList, the list of all attributes;
         cacheList, the list of cached anonymized
         dataset.
output: Result, the result of node check.
1  for attribute ∈ attributeList do
2      if attribute ∈ cacheList then
3          if cacheList[attribute] == True then
4              continue;
5          else
6              return False;
7      else
8          resultData ← generalize(dataset,
             attribute, node);
9          for class ∈ resultData do
10             if class.length > k then
11                 continue;
12             else
13                 return False;
14 return True;
```

First, in Line (2), the node check algorithm checks if the attribute that needs to be checked exists in *cacheList*. Then, if it exists, the node check algorithm will return the anonymized dataset in Line (2)–(4). Moreover, in Line (6), if the anonymized dataset in *cacheList* is non-anonymous, the node check algorithm will record that the node is non-anonymous. Conversely, if the attribute does not exist in *cacheList*, the node check algorithm will generalize the dataset and check whether each equivalent class meets the k-anonymity principle in Line (8)–(13). Finally, in Line (14), the node check algorithm marks the node as anonymous when each attribute is checked.

Furthermore, we design the path check algorithm based on the node check and data cache algorithm. The path check algorithm is responsible for checking whether the nodes on the path are anonymous. It checks the node in the middle of the path and then performs a binary search. When the node is non-anonymous, the path check algorithm adds it into the minimum heap because the predecessors of this node are non-anonymous. Therefore, the following search is in the successors, and the path check

algorithm sets *low* to *mid* + 1. If the node is anonymous, the current node is set to the possible local optimal solution. The successors of the anonymous node must also be anonymous. Only the successors need to be checked to find the optimal generalization plan. Therefore, the algorithm sets *high* to *mid* − 1. Finally, the path check algorithm repeats the above process until all nodes in the path are checked.

The outer loop algorithm traverses the anonymity lattice and finds the optimal node. The specific algorithm flow is shown in Algorithm 2.

Algorithm 2: Outer Loop of Inco-Flash

```
input : lat, the anonymity lattice.
output: resultData, the optimal anonymized
        dataset;
        node, the optimal generalization plan.
1  new heap;
2  for l ∈ lat.levels do
3      for node ∈ lat[l] do
4          if node is not marked then
5              path ← pathFind(node);
6              opi ← pathCheck(path);
7              while heap not NULL do
8                  node ← heapTop(heap);
9                  for next_i ∈ successors of node do
10                     if next_i is not marked then
11                         path ← pathFind(next_i);
12                         opi ← pathCheck(path);
13 return opi;
```

To introduce Inco-Flash clearly, we process Inco-Flash in Fig. 3 as an example. Inco-Flash starts from the 0th level and traverses all levels in the anonymity lattice. Firstly, Inco-Flash finds a path from node (0, 0, 0), which is $(0,0,0) \rightarrow (1,0,0) \rightarrow (1,1,0) \rightarrow (1,2,0) \rightarrow (1,2,1) \rightarrow (1,2,2) \rightarrow (1,2,3) \rightarrow (1,2,4)$, as shown in Line(4). Then, in Line (5), Inco-Flash uses the path check algorithm to find the optimal node. Inco-Flash firstly checks node (1, 2, 0). If (1, 2, 0) is non-anonymous, the nodes before (1, 2, 0) will be non-anonymous. Therefore, Inco-Flash adds the non-anonymous nodes to the minimum heap and checks the path from (1, 2, 1) to (1, 2, 4). Otherwise, the nodes after (1, 2, 0) will be anonymous. Then, Inco-Flash checks the path from (0, 0, 0) to (1, 1, 0) to find the optimal anonymous node. When each node in this path is marked, the top node in the minimum heap is inputted into the path find algorithm to generate a new path in Line (8)–(11). Moreover, Inco-Flash inputs the new path into the path check algorithm and updates the optimal node in Line (12). If the minimum heap is empty, Inco-Flash will traverse the next level, generate a new path and check the node of this path. Finally, when all nodes are checked, Inco-Flash will return the optimal node as the optimal generalization plan in Line (13).

In addition, Inco-Flash always calls the node check algorithm from the middle level of the anonymity lattice. Moreover, half of the nodes inputted into the node check algorithm have cached anonymized datasets, and Inco-Flash can skip the check process of these nodes. Therefore, Inco-Flash can be seen as a binary search algorithm whose time complexity is $o(m * n * \log l)$. In detail, m is the number of records, n is the number of attributes, and l is the summation of attributes generalization level. In addition, the space complexity of Inco-Flash is $o(m * n * l)$, meaning each generalized attribute is cached by Inco-Flash.

5.2 The procedure of Inco-Flash2

In the anonymized dataset, we found that the distribution of equivalent classes is highly uneven. The biggest equivalent classes are 73.3% of the anonymized dataset. It means that the dataset has been overgeneralized to satisfy the k-anonymity principle. Therefore, most of the sizes of the equivalent classes are much larger than k. To solve the overgeneralization problem, we further propose Inco-Flash2. In the node check algorithm of Inco-Flash2, the k-anonymity principle is appropriately relaxed. If the sum of the equivalent classes with sizes smaller than k does not exceed half of all equivalent classes, Inco-Flash2 will delete these equivalent classes. In addition, considering the high real-time requirements of IoT applications, the privacy-preserving algorithm does not need to find the global optimal generalization plan. A local optimal generalization plan that meets the requirement of overall metric is enough for IoT applications. Therefore, we add a termination mechanism to Inco-Flash2. In the case of meeting the requirement of overall metric, Inco-Flash2 directly terminates to reduce the time consumption.

First, the node check algorithm of Inco-Flash2 is shown in Algorithm 3.

In Line (2)–(7), same as Inco-Flash, the node check algorithm of Inco-Flash2 accesses the *cacheList* and returns the cached anonymized datasets. Then, it generalizes the datasets and gets the equivalent classes in Line (9). For each equivalent class, if its size is smaller than k, Inco-Flash2 will see it as a small equivalent class. Moreover, if the number of small equivalent classes is smaller than half of the raw datasets, Inco-Flash2 will delete these small equivalent classes in Line (10)–(12). Otherwise, Inco-Flash2 marks the node as non-anonymous in Line (14)–(16).

Then, the outer loop of Inco-Flash2 is shown in Algorithm 4.

Algorithm 3: Node Check of Inco-Flash2

```
input : node, the node needs to be checked;
        dataset, raw dataset;
        attributeList, the list of all attributes;
        cacheList, the list of cached anonymized
        dataset.
output: Result, the result of node check.
numIgnore ← 0;
for attribute ∈ attributeList do
    if attribute ∈ cacheList then
        if cacheList[attribute] == True then
            continue;
        else
            return False;
    else
        resultData ← generalize(dataset,
         attribute, node);
        for class ∈ resultData do
            if class.length > k then
                continue;
            else
                numIgnore+ = class.length;
                if numIgnore >
                 (1/2 × dataset.lentgth) then
                    return False;
return True;
```

Algorithm 4: Outer Loop of Inco-Flash2

```
input : lat, the anonymity lattice; Value, the
        generalization plan metric requirement.
output: resultData, the optimal anonymized
        dataset;
        node, the optimal generalization plan.
new heap;
for l ∈ lat.levels do
    for node ∈ lat[l] do
        if node is not marked then
            path ← pathFind(node);
            opi ← pathCheck(path);
            Value_tmp ← MetricCal(opi);
            if Value_tmp < Value then
                break;
            while heap not NULL do
                node ← heapTop(heap);
                for next_i ∈ successors of node do
                    if next_i is not marked then
                        path ← pathFind(next_i);
                        opi ← pathCheck(path);
                        if Value_tmp < Value then
                            break;
return opi
```

In Line (1)–(6), same as Inco-Flash, Inco-Flash2 uses the path find and path check algorithms to mark the nodes and find the optimal node. Then, Inco-Flash2 calculates the overall metric value of the optimal node in Line (7). If the overall metric value exceeds requirements, Inco-Flash2 will end the search process and return the present optimal node in Line (8)–(9). Finally, in Line (10)–(17), Inco-Flash2 searches the nodes in the minimum heap and applies the same termination condition for the optimal nodes.

In addition, considering the worst-case scenario, Inco-Flash2 will search the whole anonymity lattice without triggering the termination mechanism. Therefore, the time complexity of Inco-Flash2 is the same as Inco-Flash, which is $o(m * n * \log l)$. In real-world datasets, Inco-Flash2 could have lower time consumption than Inco-Flash because of the termination mechanism. Moreover, Inco-Flash2's space complexity is also $o(m * n * l)$.

6 Experiment

6.1 Dataset

This article uses three open-access datasets: the 1994 US census database (ADULT), the medical expenditure panel survey (MEPS), and the T-Drive trajectory data sample (T-Drive) [44, 45]. ADULT simulates a real IoT application dataset for evaluating privacy-preserving algorithms. MEPS is used to evaluate the privacy-preserving algorithms in short-term data. T-Drive includes a trajectory of a vehicle within a week, which is used to evaluate the privacy-preserving algorithms in vehicular IoT applications. Moreover, all generalization levels of these datasets' attributes are between 2 and 6.

6.2 Experimental environment

This paper uses a personal PC to conduct the experiments to simulate the edge computing devices with limited computing resources. We use an i7-8750H, a 2.2GHz Intel Core, and the system was a 64-bit Window10 professional 2004. Moreover, we use Python3.8 to realize Flash, Inco-Flash and Inco-Flash2. We use different k $(2 \leq k \leq 10)$ values for these algorithms and each dataset. The following experimental results are the average of the ten experiments. In terms of procedure cache, we stored anonymized data in the memory. The specific experimental results are as follows.

6.3 The total time consumption comparison of the Flash and Inco-Flash

As shown in Fig. 4, the total time consumption of Flash is more than 1 minute, and the total time consumption of Inco-Flash is within 25 seconds. Moreover, the total consumption of Inco-Flash is only 34.3% of that of Flash. Flash needs to generalize whole datasets to check whether the node satisfies the k-anonymity principle, resulting in high time consumption in node check and path check algorithms. In addition, Flash cannot use cached anonymized datasets in the node check algorithm. As a result, the total time consumption of Flash is much higher than Inco-Flash. Inco-Flash applies the generalization plan to each attribute during the node check algorithm to save time. Moreover, Inco-Flash uses more anonymized datasets than Flash, decreasing time consumption. In addition, as shown in Figs. 5 and 6, the time consumption of Flash and Inco-Flash on both MEPS and T-Drive. It means that Inco-Flash can also optimize the time consumption in short-term data.

6.4 The overall metric comparsion of Flash and Inco-Flash

This section evaluates the overall metric value of Inco-Flash and Flash in ADULT. As shown in Fig. 7, Inco-Flash and Flash have the same overall metric value when $k = 2, 7, 8, 9, 10$. The optimal anonymized datasets generated by Inco-Flash with these k values are the same as Flash's. However, when $k = 3, 4, 5, 6$, Inco-Flash has a lower overall metric than Flash. It is because that Flash only focuses on the disclosure risk but ignores the information loss. In conclusion, the optimal anonymized datasets generated by Inco-Flash have the better or same overall metric as that of Flash. Furthermore, compared with Flash, Inco-Flash can effectively balance the information loss and disclosure risk.

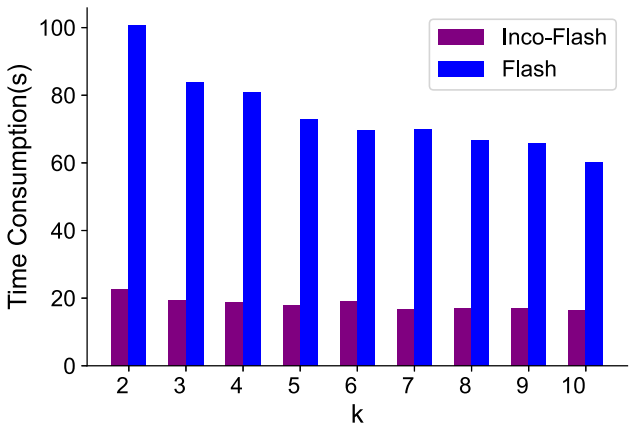

Fig. 4 Total running time of Flash and Inco-Flash on ADULT

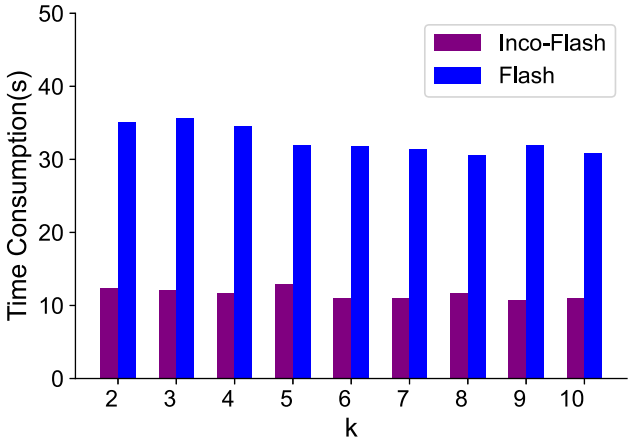

Fig. 5 Total running time of Flash and Inco-Flash on MEPS

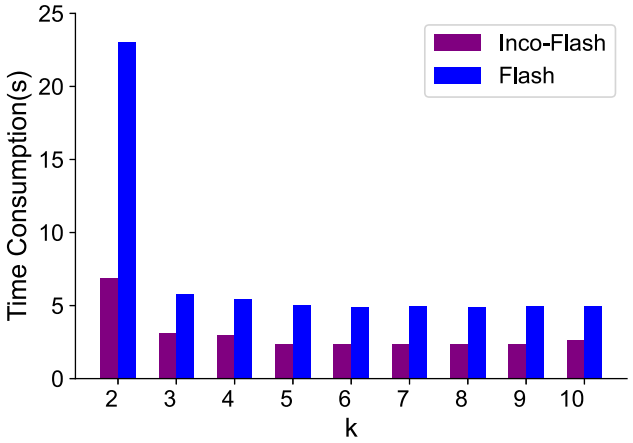

Fig. 6 Total running time of Flash and Inco-Flash on T-Drive

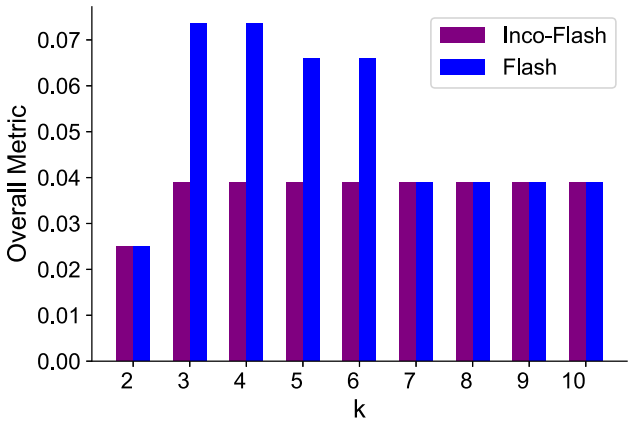

Fig. 7 Overall metric value of Flash and Inco-Flash on ADULT

6.5 Details of the algorithm time consumption

We also recorded the time consumption of each part of Flash and Inco-Flash, including the path find, node check, data cache, and path check algorithms. In detail, the path check algorithm calls the node check and data cache algorithms. So the time consumption of the path check algorithm includes that of the node check and data cache algorithms. Moreover, the path find and data cache algorithms have fewer calls and low time complexity, so their time consumption is low. Therefore, we will hide the time consumption of data cache and path find algorithm in the following experimental results. Figure 8 is the time consumption of each algorithm on the ADULT dataset. Figure 9 is the time consumption of each algorithm on the MEPS dataset.

Experimental results of the ADULT dataset show that most time of Flash and Inco-Flash is spent on the node check algorithm. Moreover, the time consumption of the node check algorithm is close to the total time consumption. It is because the node check algorithm operates many times and the time consumption of the path find algorithm is low. In addition, the time consumption of the node check algorithm and path check algorithm is close, meaning that the path check algorithm consumes most operation time on node check and the time consumption of data cache and minimum heap operations is low. Therefore, the time consumption of the node check algorithm always accounts for more than 80% of the total time consumption of Flash and Inco-Flash. In addition, since the path check algorithm calls the node check algorithm, its time consumption is the sum of the time consumption of the node check and path find algorithms. Moreover, in MEPS dataset, Flash and Inco-Flash consumed almost all the time on the node check algorithm. Therefore, we find that the number of attributes impacts the path check process, but the records number only affects the time consumption of the node check algorithm.

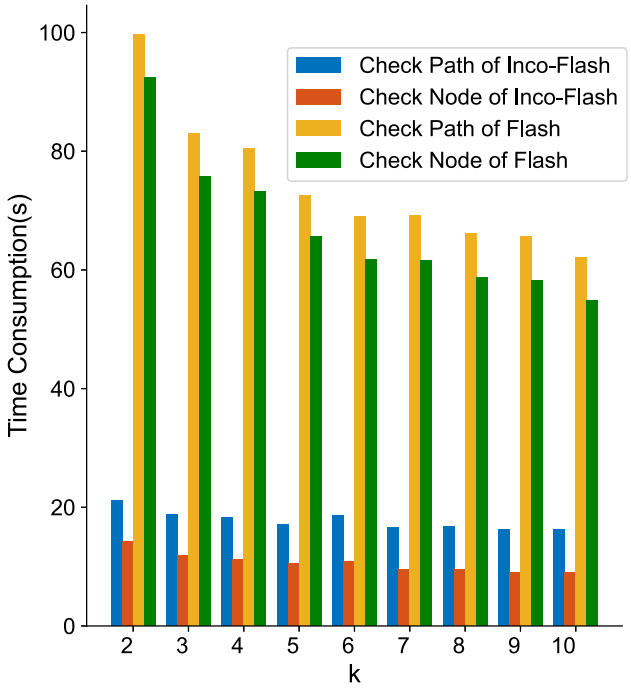

Fig. 8 Time spent in each part of Flash and Inco-Flash on ADULT

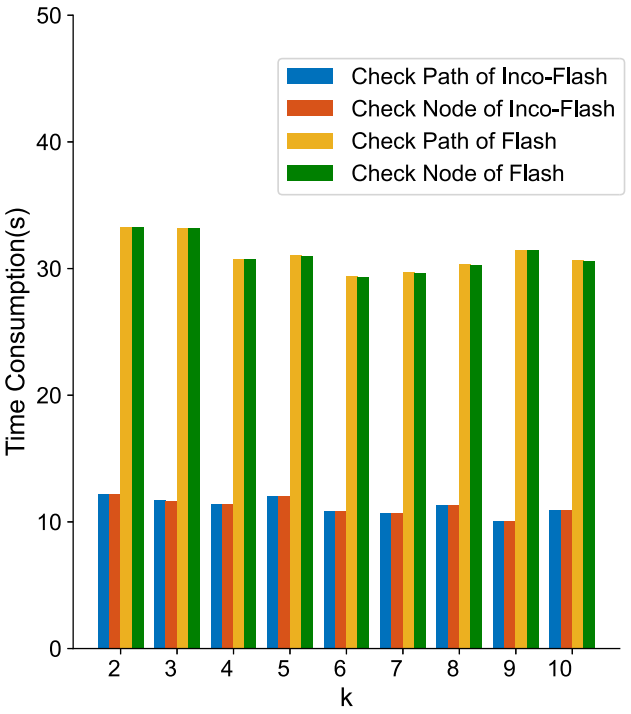

Fig. 9 Time spent in each part of Flash and Inco-Flash on MEPS

6.6 The total time consumption comparison of Inco-Flash and Inco-Flash2

Inco-Flash2 needs an overall metric requirement for the termination mechanism. Therefore, we calculated the overall metric values of the local optimal generated by the Inco-Flash algorithm on ADULT. Figure 10 shows the specific distribution of the overall metric values.

We take the 90% quantile (0.0309) in the overall metric values as the overall metric requirement of the Inco-Flash2 algorithm. It means that the anonymized dataset generated by Inco-Flash2 is at least better than 90% of other generalization plans. Then, we conduct experiments on Inco-Flash2 on the ADULT dataset and compare it with Inco-Flash. As shown in Fig. 11, the time consumption of Inco-Flash2 is within 4 seconds. When $k = 2$ and $k = 3$, Inco-Flash2 only searches 9 anonymous nodes. Moreover, when $k > 3$, Inco-Flash2 only searches for 6 anonymous nodes, reducing the time consumption. Finally, the time

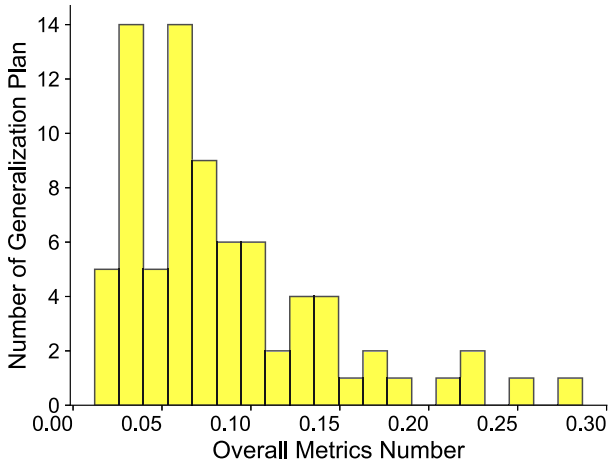

Fig. 10 Distribution of overall metrics on the ADULT dataset

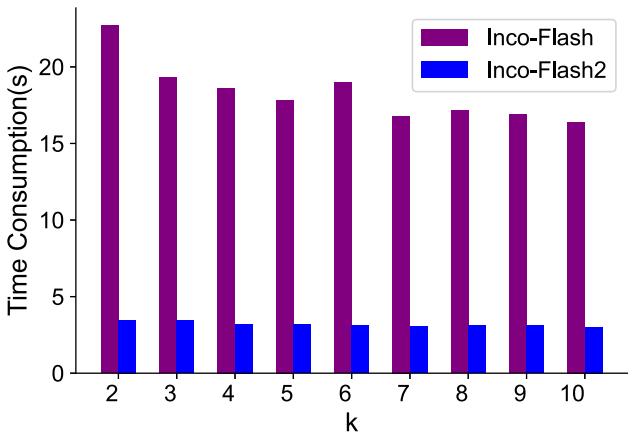

Fig. 11 Total time consumption of Inco-Flash and Inco-Flash2

consumption of Inco-Flash2 is less than 25% of that of Inco-Flash.

6.7 The equivalent class distributions of Inco-Flash and Inco-Flash2

To verify the solvation of the overgeneralization problem, we record the equivalent class distribution of Inco-Flash on the ADULT dataset when $k = 10$. The distribution of the equivalent classes is shown in Fig. 12.

As shown in Fig. 12, the biggest equivalent class accounts for 73.3% of the anonymized dataset. It means the overgeneralization problem appears in Flash and Inco-Flash. Figure 13 is the equivalent class distribution of the anonymized dataset generated by Inco-Flash2. The equivalent class distribution in this anonymized dataset is more balanced than that generated by Inco-Flash and Flash. In detail, most of the equivalent classes' sizes are close to the k value. Therefore, the overgeneralization problem is solved by Inco-Flash2.

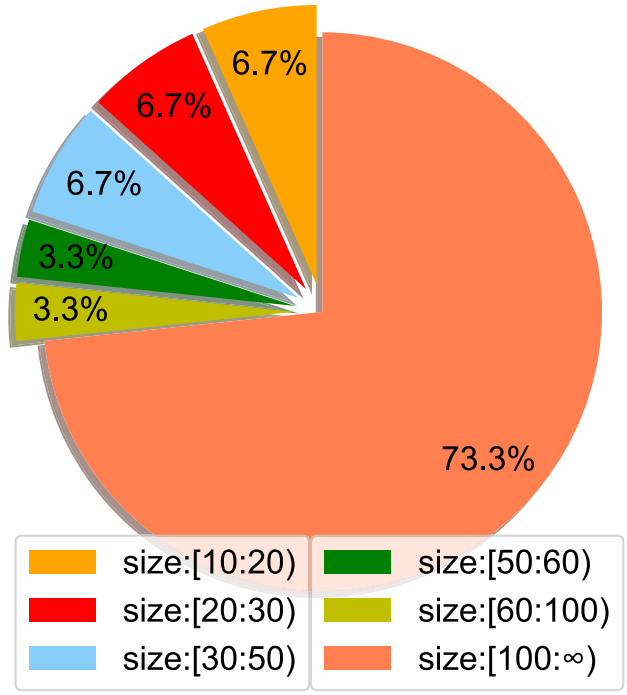

Fig. 12 Equivalent class distribution of Inco-Flash and Flash

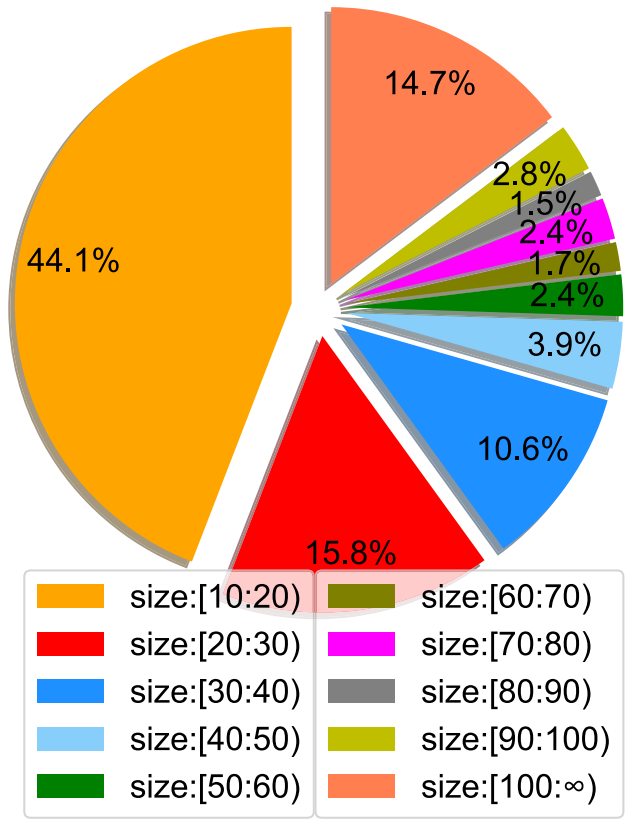

Fig. 13 equivalent class distribution of Inco-Flash2

6.8 The accuracy of IoT application trained by Inco-Flash2 generated dataset

The deep learning models are widely used in IoT applications, such as big data analysis and environment perception [46]. Therefore, we also validate the usability of the proposed algorithms in a deep neural model to simulate the IoT applications' environment. The deep neural model has 50 hidden neurons, and its activation function is *relu*. We use the ADULT anonymized dataset generated by Inco-Flash2 to classify the individuals' incomes. If the income of a record is smaller than 50k, the model's output is 0. Otherwise, the output of it is 1. Then we use the Pytorch to realize this model and train it for 300 epochs. Moreover, each dataset used in the training process is randomly divided by 20% and 80%, in which 20% is the validation dataset, and 80% is the training dataset. We compare the accuracy of the trained model with the anonymized dataset and the raw dataset, which is the ratio between the number of correct classification results and the number of records. The detailed accuracies are shown in Fig. 14.

As shown in Fig. 14, the accuracies of these two models are similar, and the differences between them are below 6%. Therefore, the information loss of Inco-Flash2 is controlled in an acceptable range, ensuring the usability of the dataset for IoT applications. Moreover, the model trained by the anonymized dataset has higher accuracy than the one trained by the raw dataset. It means that the small equivalent classes deleted by Inco-Flash2 include abnormal records. In raw datasets, the classification model sees these abnormal records as noise, which may decrease the classification accuracy. Therefore, by deleting small equivalent classes, Inco-Flash2 can effectively filter out abnormal records and improve the accuracies of IoT applications. In addition, the models trained by anonymized datasets with different k values have the same accuracy, which is 83.15%. It further indicates that Inco-Flash2 generalizes the attributes unrelated to the individuals' incomes, simplifying the classification problem. So that the classification model can easily converge to the point with a higher accuracy based on all anonymized datasets with different k values.

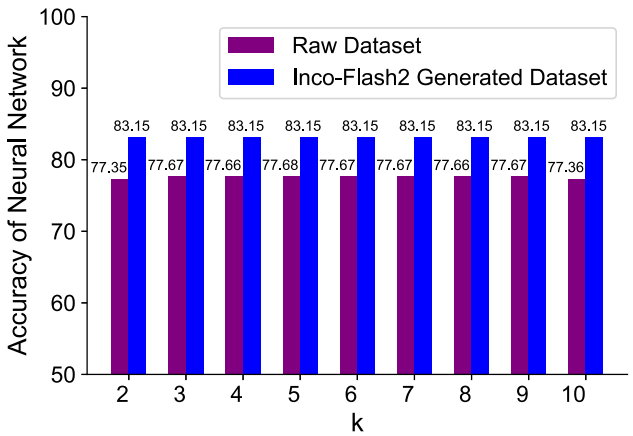

Fig. 14 Accuracies of deep neural model

7 Conclusion

This article focuses on the privacy-preserving algorithms for IoT applications in virtualization and edge computing environments. To satisfy the real-time and fine-grained datasets requirements, we propose two Inco-Flash and Inco-Flash2 to balance the disclosure risk and information loss. Based on procedure cache and small equivalent classes deletion scheme, Inco-Flash and Inco-Flash2 effectively decrease the time consumption and ensure the quality of the datasets. The experiments demonstrate that Inco-Flash only consumes 34.3% time of Flash. Moreover, the overgeneralization problem is solved by Inco-Flash2. However, our privacy-preserving algorithms still need generalization trees to construct the anonymity lattice, which some IoT applications cannot satisfy. In addition, the performance of proposed algorithms isn't evaluated in real virtualization and edge computing environments. Therefore, one promising future direction is implementing a generalization trees generation method in our privacy-preserving algorithms and evaluating our algorithms for real IoT applications in virtualization and edge computing environments.

Acknowledgements We are thankful to Dr. Weizhe Zhang School of Computer Science and Technology, Harbin Institute of Technology for his valuable contribution towards the proposed work. Dr. Weizhe Zhang is the corresponding author of this article.

Author contributions All the authors contributed equally in conducting the research. The authors have participated in conception and design, or analysis and interpretation of the data, drafting the article or revising it critically for important intellectual content, approval of the final version.

Funding This work was supported in part by the Key-Area Research and Development Program of Guangdong Guangdong Province (2020B0101360001), the Fundamental Research Funds for the Central Universities (Grant No. HIT.OCEF.2021007) and the Shenzhen Science and Technology Research and Development Foundation (JCYJ20190806143418198).

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing interests The author declares that he has no competing interests.

References

1. Morabito, R.: Virtualization on internet of things edge devices with container technologies: a performance evaluation. IEEE Access **5**, 8835–8850 (2017). https://doi.org/10.1109/ACCESS.2017.2704444
2. Almohaimeed, A., Gampa, S., Singh, G.: Privacy-preserving iot devices. In: 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp. 1–5 (2019). https://doi.org/10.1109/LISAT.2019.8817349
3. Jiang, X., Lora, M., Chattopadhyay, S.: An experimental analysis of security vulnerabilities in industrial IoT devices. ACM Trans. Internet Technol. **20**(2), 16–11624 (2020)
4. Sweeney, L.: Simple demographics often identify people uniquely. Carnegie mellon university. Journal contribution. (2018). https://doi.org/10.1184/R1/6625769.v1
5. Sweeney, L., Yoo, J.S.: De-anonymizing South Korean resident registration numbers shared in prescription data, Technol. Sci., pp. 1–27, Sep. (2015), 2015092901.
6. Kohlmayer, F., Prasser, F., Eckert, C., Kemper, A., Kuhn, K.A.: Flash: Efficient, stable and optimal k-anonymity. In: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, pp. 708–717 (2012)
7. Keshavamurthy, B.N., Khan, A.M., Toshniwal, D.: Privacy preserving association rule mining over distributed databases using genetic algorithm. Neural Comput. Appl. **22**(1), 351–364 (2013). https://doi.org/10.1007/s00521-013-1343-9
8. Zigomitros, A., Casino, F., Solanas, A., Patsakis, C.: A survey on privacy properties for data publishing of relational data. IEEE Access **8**, 51071–51099 (2020)
9. Sweeney, L.: K-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
10. Javadpour, A., Wang, G., Rezaei, S.: Resource management in a peer to peer cloud network for IoT. Wirel. Pers. Commun. **115**(3), 2471–2488 (2020). https://doi.org/10.1007/s11277-020-07691-7
11. Javadpour, Amir, Wang, Guojun: cTMvSDN: improving resource management using combination of Markov-process and TDMA in software-defined networking. J. Supercomput. **78**(3), 3477–3499 (2022). https://doi.org/10.1007/s11227-021-03871-9
12. Mirmohseni, S.M., Tang, C., Javadpour, A.: Using Markov learning utilization model for resource allocation in cloud of thing network. Wirel. Pers. Commun. **115**(1), 653–677 (2020). https://doi.org/10.1007/s11277-020-07591-w
13. Javadpour, A., Abharian, S.K., Wang, G.: Feature selection and intrusion detection in cloud environment based on machine learning algorithms. In: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), pp. 1417–1421 (2017). https://doi.org/10.1109/ISPA/IUCC.2017.00215
14. Pachilakis, M., Chariton, A.A., Papadopoulos, P., Ilia, P., Degkleri, E., Markatos, E.P.: Design and implementation of a compressed certificate status protocol. ACM Trans. Internet Technol. **20**(4), 34–13425 (2020)
15. Anagnostopoulos, N.A., Ahmad, S., Arul, T., Steinmetzer, D., Hollick, M., Katzenbeisser, S.: Low-cost security for next-generation IoT networks. ACM Trans. Internet Technol. **20**(3), 30–13031 (2020)
16. Zhang, W., He, H., Zhang, Q., Kim, T.-H.: PhoneProtector: protecting user privacy on the android-based mobile platform. Int. J. Distrib. Sensor Netw. **10**(2), 282417 (2014)
17. Zhang, W., Li, X., Xiong, N., Vasilakos, A.V.: Android platform-based individual privacy information protection system. Pers. Ubiquit. Comput. **20**(6), 875–884 (2016)
18. Sengupta, B., Li, Y., Bu, K., Deng, R.H.: Privacy-preserving network path validation. ACM Trans. Internet Technol. **20**(1), 5–1527 (2020)
19. Liu, X., Yang, G., Susilo, W., Tonien, J., Liu, X., Shen, J.: Privacy-preserving multi-keyword searchable encryption for distributed systems. IEEE Trans. Parallel Distrib. Syst. **32**(3), 561–574 (2020)
20. Tao, Y., Pei, J., Li, J., Xiao, X., Yi, K., Xing, Z.: Correlation hiding by independence masking. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), pp. 964–967 (2010)
21. Terrovitis, M., Mamoulis, N., Liagouris, J., Skiadopoulos, S.: Privacy preservation by disassociation. Proc. VLDB Endow. **5**(10), 944–955 (2012)
22. Lin, K., Chen, M.: On the design and analysis of the privacy-preserving SVM classifier. IEEE Trans. Knowl. Data Eng. **23**(11), 1704–1717 (2011)
23. Ben Haj Frej, M., Dichter, J., Gupta, N.: Light-weight accountable privacy preserving (lapp) protocol to determine dishonest role of third party auditor in cloud auditing. In: 2018 IEEE International Conference on Consumer Electronics (ICCE), pp. 1–6 (2018)
24. Frej, M.B.H., Dichter, J., Gupta, N.: Comparison of privacy-preserving models based on a third-party auditor in cloud computing. In: 2019 IEEE Cloud Summit, pp. 86–91 (2019)
25. Zhang, Y., Li, S.: Privacy preserving birch algorithm under differential privacy. In: 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), pp. 48–53 (2017)
26. Zhang, Y., Zhong, S.: A privacy-preserving algorithm for distributed training of neural network ensembles. Neural Comput. Appl. **22**(1), 269–282 (2013). https://doi.org/10.1007/s00521-012-1000-8
27. Reza, K.J., Islam, M.Z., Estivill-Castro, V.: Privacy protection of online social network users, against attribute inference attacks, through the use of a set of exhaustive rules. Neural Comput. Appl. (2021). https://doi.org/10.1007/s00521-021-05860-8
28. Wu, X., Zhang, Y., Wang, A., Shi, M., Wang, H., Liu, L.: MNSSp3: medical big data privacy protection platform based on internet of things. Neural Comput. Appl. (2020). https://doi.org/10.1007/s00521-020-04873-z
29. Nikolaidis, S., Refanidis, I.: Privacy preserving distributed training of neural networks. Neural Comput. Appl. **32**(23), 17333–17350 (2020). https://doi.org/10.1007/s00521-020-04880-0
30. Mahanan, W., Chaovalitwongse, W.A., Natwichai, J.: Data privacy preservation algorithm with k-anonymity. World Wide Web **24**(5), 1551–1561 (2021). https://doi.org/10.1007/s11280-021-00922-2
31. Kessler, S., Hoff, J., Freytag, J.-C.: SAP HANA goes private: from privacy research to privacy aware enterprise analytics. Proc. VLDB Endow. **12**(12), 1998–2009 (2019). https://doi.org/10.14778/3352063.3352119
32. Zhao, S., Li, F., Li, H., Lu, R., Ren, S., Bao, H., Lin, J.-H., Han, S.: Smart and practical privacy-preserving data aggregation for fog-based smart grids. IEEE Trans. Inf. Forensics Sec. **16**, 521–536 (2020)
33. Pang, H., Shen, J., Krishnan, R.: Privacy-preserving similarity-based text retrieval. ACM Trans. Internet Technol. **10**(1), 1–39 (2010)
34. Palanisamy, B., Liu, L., Zhou, Y., Wang, Q.: Privacy-preserving publishing of multilevel utility-controlled graph datasets. ACM Trans. Internet Technol. **18**(2), 1–21 (2018)

35. Qian, J., Li, X.-Y., Zhang, C., Chen, L.: De-anonymizing social networks and inferring private attributes using knowledge graphs. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016). https://doi.org/10.1109/INFOCOM.2016.7524578
36. Zhu, D., Zhang, F., Wang, S., Wang, Y., Cheng, X., Huang, Z., Liu, Y.: Understanding place characteristics in geographic contexts through graph convolutional neural networks. Ann. Am. Assoc. Geogr. **110**(2), 408–420 (2020). https://doi.org/10.1080/24694452.2019.1694403
37. Chen, F., Fu, Z., Yang, Z.: Wind power generation fault diagnosis based on deep learning model in internet of things (IoT) with clusters. Clust. Comput. **22**(6), 14013–14025 (2019). https://doi.org/10.1007/s10586-018-2171-6
38. Etemadi, M., Ghobaei-Arani, M., Shahidinejad, A.: A cost-efficient auto-scaling mechanism for IoT applications in fog computing environment: a deep learning-based approach. Clust. Comput. **24**(4), 3277–3292 (2021). https://doi.org/10.1007/s10586-021-03307-2
39. Hilal, A.M., Alohali, M.A., Al-Wesabi, F.N., Nemri, N., Alyamani, H.J., Gupta, D.: Enhancing quality of experience in mobile edge computing using deep learning based data offloading and cyberattack detection technique. Clust. Comput. (2021). https://doi.org/10.1007/s10586-021-03401-5
40. Zhang, W., Yadav, R., Tian, Y.-C., Tyagi, S.K.K.S., Eelgendy, I.A., Kaiwartya, O.: Two-phase industrial manufacturing service management for energy efficiency of data centers. IEEE Trans. Ind. Inf. **18**(11), 7525–7536 (2022)
41. Yadav, R., Zhang, W., Elgendy, I.A., Dong, G., Shafiq, M., Laghari, A.A., Prakash, S.: Smart healthcare: Rl-based task offloading scheme for edge-enable sensor networks. IEEE Sens. J. **21**(22), 24910–24918 (2021)
42. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: 21st International Conference on Data Engineering (ICDE'05), pp. 217–228 (2005)
43. El Emam, K.: Methods for the de-identification of electronic health records for genomic research. Genome Med. **3**(4), 25 (2011). https://doi.org/10.1186/gm239
44. Yuan, J., Zheng, Y., Xie, X., Sun, G.: Driving with knowledge from the physical world. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '11, pp. 316–324. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2020408.2020462
45. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: Driving directions based on taxi trajectories. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. GIS '10, pp. 99–108. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1869790.1869807
46. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M.: Deep learning for IoT big data and streaming analytics: a survey. IEEE Commun. Surveys Tutorials **20**(4), 2923–2960 (2018). https://doi.org/10.1109/COMST.2018.2844341

Chen Ling received the B.S. degree in Software Engineering from Beijing University of Technology, Beijing, China in 2018. He is currently working toward the Ph.D. degree in Computer Science and Technology with the School of Computer Science and Technology, Harbin Institute of Technology, Xidazhi Street, Harbin, China. His research interests include edge computing, vehicular network and IoT application.

Weizhe Zhang (Senior Member, IEEE) received B.Eng, M.Eng and Ph.D. degree of Engineering in computer science and technology in 1999, 2001 and 2006 respectively from Harbin Institute of Technology. He is currently a professor in the School of Computer Science and Technology at Harbin Institute of Technology, China, and director in the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He has published more than 100 academic papers in journals, books, and conference proceedings.

Hui He (Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1997, 1999, and 2006, respectively, all in computer science and technology. She is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology. Her current research interests include distributed computing, IoT and big data analysis.