Towards Client-Efficient Privacy-Preserving Multi-Party Multi-Data Sorting Scheme

Shuai Shang, Xiong Li, Rongxing Lu, *Fellow, IEEE*, Xiaosong Zhang

***Abstract*—Edge-supported industrial Internet of Things (IIoT) has recently received significant attention, where edge computing can not only provide data storage locally and almost real-time data process, but also reduce the communication overhead and computational costs of IoT devices, which can greatly improve the service quality of IIoT applications. Privacy-preserving range query is one of the most important functional requirements for edge-supported IIoT. Recently, some privacy-preserving range query solutions have been proposed in different fields. However, most of them only support single-dimensional range query, which are inefficient for the requirement of multi-dimensional range query. To address this challenge, we propose a privacy-preserving multi-dimensional range query scheme for edge-supported IIoT in this paper, called Edge-PPMRQ. In Edge-PPMRQ, we design a novel ranges division algorithm to support multi-dimensional range query. Besides, Edge-PPMRQ also supports the range queries for continuous, discontinuous and arbitrary boundary ranges. The detailed security analysis proves that Edge-PPMRQ is really privacy-preserving for the query ranges, the query result and the sensed data of IoT devices. Furthermore, extensive comparison experiments also illustrate that Edge-PPMRQ is remarkably efficient in communication and computation.**

***Index Terms*—Range query, multi-dimensional, industrial Internet of Things (IIoT), edge computing, privacy-preserving.**

I. Introduction

With the significant advances of information technology, Internet of Things (IoT) is widely applied in different fields, e.g., smart grids [1], intelligent parking [2], smart homes [3] and indoor navigation [4], which makes our daily life more and more convenient. Especially, with the applications of IoT in industrial fields, the so called Industrial Internet of Things (IIoT) [5] emerges, which can greatly improve work efficiency and reduce resource consumption. As an emerging computing technology, edge computing [6] can store and process data near the IoT devices, so it can not only greatly reduce the communication load and computational costs of IoT devices to extend their life cycle, but also achieve almost real-time data processing. Therefore, edge computing is very suitable for supporting IIoT to improve its work efficiency. In edge-supported IIoT, a large number of sensors are deployed in industrial environment to collect different types of real-time data, and the edge server is capable to receive and process the sensed data of IIoT sensors locally. The results of data processing can help the industrial devices to make precise decision and control which can evidently improves the work efficiency and reliability of industrial devices. For edge-supported IIoT applications, range query is one of the most significant functional requirements. For example, in a factory, different types of industrial sensors are deployed in the factory to collect real-time environment data, such as water volume, power consumption, pressure and temperature. Based on how many sensors, whose sensed data exceeds the threshold values, the manager of the factory can determine whether water volume and temperature value are in normal ranges or not. Meanwhile, if the data is gathered by other parties, such as adversaries, the detailed running status of the factory will be leaked to unauthorized entities, i.e. privacy preservation [7] is a critical problem we should consider in range query. For example, the query user doesn't want to leak any information about its query range and query result. At the same time, in order to achieve range query more efficiently, the query user may want to perform multi-dimensional range query, i.e., the query user can get query results of different types of data by one query request.

A. Related Work

In recent years, a lot of privacy-preserving range query solutions have been proposed in different fields, e.g., wireless sensor networks (WSN) [8–11], cloud computing [12–15], vehicle sensing systems [16], P2P networks [17] and IoT [18–23]. In WSN, Yi et al. [8] proposed an efficient scheme to process range queries in two-tier sensor networks, which supports the functions of privacy and integrity preservation. Tsou et al. [9] proposed an efficient and secure anonymous range query scheme for WSN. Zeng et al. [10] proposed an energy-efficient range query scheme in 2017, which can support multi-dimensional range query. Liu et al. [11] designed a spatial range aggregation query scheme for dynamic sensor networks supporting privacy-preserving. In cloud computing, Li et al. [12] proposed a range query protocol in cloud based on PBtree data structure. It not only achieves strong privacy-preserving but also supports real-time queries. Xu et al. [13] designed a lightweight scheme supporting both single-dimensional and multi-dimensional range queries, and it can protect the data privacy and the integrity of query results. Li et al. [14] proposed a secure scheme of multi-dimensional range query. It not only achieves sub-linear search

S. Shang and X. Li are with the Institute for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: shshang180@gmail.com, lixiong@uestc.edu.cn).

R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, E3B 5A3, Canada (e-mail: RLU1@unb.ca)

X. Zhang is with the Institute for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China and with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, Guangdong 518040, China (e-mail: johnsonzxs@uestc.edu.cn).

efficiency, but also is secure in known-background model. Liang et al. [15] designed a multi-source scheme with order-preserving encryption for eHealth systems, which supports range queries of different patients. In vehicle sensing systems, Peng et al. [16] introduced an efficient range query scheme for secure vehicle sensing systems supporting location privacy-preservation. In P2P networks, Lim et al. [17] proposed an efficient range query scheme in mobile P2P, which includes two phases, i.e. query distribution phase and monitoring phase. In the first phase, it prunes the peers that are impossible to be included in the query result. In the second phase, their scheme updates the query result incrementally. The scheme remarkably reduces the cost of processing a continuous range query in mobile P2P network environments. In IoT applications, Li et al. [18] proposed a multi-attribute aggregation query mechanism in the context of edge computing, where an energy-aware IR-tree is constructed to process query requests in single edge network, and an edge node routing graph is established to facilitate query processing for marginal smart devices contained in contiguous edge networks. Djellabi et al. [19] proposed a scheme for efficient range queries in IoT. It adopts a data distribution model based on both consistent and order-preserving hash to efficiently handle the range queries. Wan et al. [20] proposed a multi-dimensional data indexing scheme, which is energy and time efficient. Mahdikhani et al. [21] presented an efficient and privacy preservation single-dimensional range query solution based on bloom filter in fog-enhanced IoT, which supports continuous and discontinuous range queries, and achieves $(n+|E|)\cdot\log n$-bit communication efficiency. In the same year, Mahdikhani et al. [22] proposed a single-dimensional privacy-preserving range query scheme in fog-based IoT, which achieves $O(\log^3 n)$ communication efficiency. But the upper and lower bounds of the query ranges of their scheme must be the powers of 2, which is inconvenient for arbitrary boundary query ranges. In 2021, Mahdikhani et al. [23] presented a privacy-preserving scheme using reduced paths, which employs a symmetric homomorphic encryption (SHE) to encrypt the reduced paths and achieves $O(\log^2 n)$ communication efficiency. Although the scheme is computationally efficient in fog node, it is at the cost of a large amount of computational overhead for IoT devices, which contraries to the original intention of the fog/edge computing applications.

Although many range query solutions have been proposed for different scenarios, most of the schemes just support the single-dimensional range query, i.e., the query user can query only single kind of sensed data by one query request. In industrial field, different kinds of data is usually needed to make intelligent decision. For example, in industrial environment, a manager needs different kinds of data collected from a variety of sensors, such as temperature, operation speed and power consumption, to determine if the devices are running normally. Consequently, in such scenarios, the query user has to launch many query requests to achieve multi-dimensional range query, i.e., the query user needs launch m query requests to get m dimensions sensed data. It not only causes the waste of time and bandwidth, but also results in the problem of response delay. However, a multi-dimensional range query scheme can solve all the challenges mentioned above. Therefore, in the paper, we focus on the design of a multi-dimensional range query scheme with privacy preservation in edge-supported IIoT.

B. Our Contributions

To achieve multi-dimensional range query, we propose the ranges division algorithm to process multi-dimensional query ranges. Based on the algorithm, we propose an efficient privacy-preserving multi-dimensional range query scheme for edge-supported IIoT, called Edge-PPMRQ. Concretely, the main contributions are as below:

1) A ranges division algorithm is designed to divide multi-dimensional query ranges into the corresponding sub-ranges. Then the sub-ranges are mapped into a group of bloom filters, through which, multi-dimensional query ranges are integrated into one query request instead of multiple query requests.
2) Based on the ranges division algorithm together with OU encryption and bloom filter, Edge-PPMRQ not only realizes privacy-preserving multi-dimensional range query by one request, but also supports continuous, discontinuous and arbitrary boundary range queries for any dimensions' data.
3) The security analysis shows that Edge-PPMRQ guarantees the privacy of the query ranges, the query result and the sensed data of IoT devices.
4) Extensive experiments are performed to evaluate the performance of Edge-PPMRQ and related schemes, and the results demonstrate that Edge-PPMRQ is efficient in communication and computation.

C. Organization of the paper

The remaining parts of this paper are arranged as follows. In section II, the preliminary knowledge is introduced. Section III describes our system model, security model and the design goals. The proposed privacy-preserving multi-dimensional range query scheme for edge-supported IIoT and the corresponding security analysis are described in section IV and V, respectively. Section VI evaluates the communication and computation performance of Edge-PPMRQ by comparing with other related schemes. Finally, section VIII concludes the full paper.

II. Preliminaries

In this section, we briefly introduce two preliminaries used in Edge-PPMRQ, i.e., bloom filter data structure [24] and OU encryption [25].

A. Bloom Filter Data Structure

Bloom filter (BF) [24] is a kind of data structure composed of a n-bits binary vector and k independent hash functions. It can be used to check whether an element is in a set or not. To achieve this mission, the vector is set to all 0 initially. For a set of integers $I = \{I_1, I_2, ..., I_s\}$, k hash functions $H_1, H_2, ..., H_k$ are called to compute $H_i(I_j) \in [1, n]$, where $i \in [1, k]$ and $j \in [1, s]$. Then all the $H_i(I_j)$-th bits in the

vector are set to 1. Until now, the set I is mapped into the BF. To check if an element e is in set I, we compute the values $H_i(e)$, where $i \in [1, k]$. If all the $H_i(e)$-th bits in the BF are 1, it can be confirmed that e is in the set I. Otherwise, e is not in the set. Since only hash operations are used in BF, it is very easy to be implemented on hardware at a high speed. Compared with other methods solving the same problem, BF costs much less storage space, insert time and query time. However, due to the fact that BF is a probability-based data structure, there exists false positive rate in BF. Fortunately, based on the relationship among the false positive rate P_f, the vector's length n, the number of inserted elements s and the number of hash functions k: $P_f = (1 - (1 - \frac{1}{n})^{ks})^k$, P_f can be constrained within an acceptable range by adjusting n, s and k. In order to understand BF better, an example is given here. Suppose that a set $I = \{11, 12, ..., 20\}$ is mapped into a 1000-bit length bloom filter by using 3 hash functions H_1, H_2 and H_3, whose output ranges are all [1, 1000]. Firstly, the hashed values $H_1(11), H_2(11), H_3(11), H_1(12), H_2(12), H_3(12), ..., H_1(20)$, $H_2(20)$, $H_3(20)$ are computed. Then, the $H_1(11)$-th, $H_2(11)$-th, $H_3(11)$-th, ..., $H_3(20)$-th bits in the vector are set to 1. When we check if 5 and 20 are the elements of I, $H_1(5), H_2(5), H_3(5), H_1(20), H_2(20), H_3(20)$ are computed to gather the values of the corresponding positions in the vector. Obviously, all the $H_1(20), H_2(20), H_3(20)$-th bits are 1, while the $H_1(5), H_2(5), H_3(5)$-th bits are 0. So, we can confirm that 20 is in I while 5 is not. For more detailed information about BF, please refer to [24].

B. OU Encryption

OU encryption is extensively used in privacy preservation applications because of its homomorphic properties. It consists of three algorithms, i.e., key generation, encryption and decryption. We describe this cryptosystem as below:

1) *KeyGeneration*: Given a security parameter κ, two large prime numbers p and q are chosen with the same length κ. A function $L(x)$ is defined as $L(x) = (x - 1)/p$. Then, $n = p^2q$ is calculated and $g \in Z_n^*$ is chosen, which satisfies that the order of g^{p-1} mod p^2 is p. Additionally, h is computed as $h = g^n$ mod n. Finally, the public key and private key of the system are $pk = (n, g, h, \kappa)$ and $sk = (p, q)$, respectively.
2) *Encryption*: Given a plaintext m, $0 \leq m \leq 2^{\kappa-1}$, a random number $r \in Z_n$ is selected. Then, m can be encrypted to a ciphertext C as $C = E(m) = g^m h^r$ mod n.
3) *Decryption*: For a ciphertext C, $C_p = C^{p-1}$ mod p^2 and $g_p = g^{p-1}$ mod p^2 are computed, and the corresponding plaintext m can be decrypted as $m = L(C_p)/L(g_p)$ mod p.

OU encryption is a homomorphic encryption algorithm supporting additive homomorphism. Given two plaintext-ciphertext pairs (m_1, C_1) and (m_2, C_2), where $C_1 = E(m_1)$ and $C_2 = E(m_2)$, we have $C_1 \cdot C_2 = E(m_1) \cdot E(m_2) = E(m_1 + m_2)$ and $C_1^{m_2} = E(m_1 \cdot m_2)$, according to the homomorphic properties.

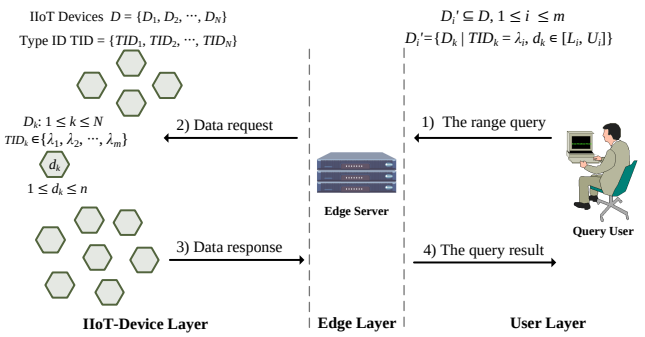


Fig. 1: System model of Edge-PPMRQ

III. Models and Design Goals

In this section, we describe the system model, security model and design goals of Edge-PPMRQ.

A. System Model

There are three types of entities involved in Edge-PPMRQ, i.e., a group of IoT devices $D = \{D_1, D_2, ..., D_N\}$, an edge server and a query user, as shown in the Fig. 1.

1) IoT devices $D = \{D_1, D_2, ..., D_N\}$: A group of N IoT sensors. They are deployed in industrial environment to collect m-dimensional data, i.e., m different types of data, such as water temperature, pressure, power consumption, etc. For each sensor, it first senses the data of the corresponding dimension in the environment, and transfers the sensed data to an edge server subsequently. In reality, the sensed data is not always integers, e.g., 4.957. To handle range query for such data, 4.957 can be transformed into an integer 4957 by multiplying 1000. Without loss of generality, we assume that in edge-PPMRQ, the sensed data d_k of the IoT device D_k is an integer within the range of $[1, n]$.
2) Edge server: An edge server has more powerful capabilities of storage, communication and computation than IoT devices. It bridges IoT devices and a query user to relieve the computational costs of IoT devices and responses to the query user at an almost real-time manner. During a range query, it receives the query request from a query user, and processes the sensed data from IoT devices accordingly to generate the ciphertexts of query results $C = \{C_1.C_2, ..., C_m\}$. Finally, C are returned to the query user.
3) Query user: A query user can directly launch a multi-dimensional range query request to edge server. For example, a query user wants to know how many IoT devices of dimension i, whose sensed data is within the range $[L_i, U_i]$, where $1 \leq i \leq m$ and $1 \leq L_i \leq U_i \leq n$. After receiving the ciphertexts of the query result from the edge server, the query user recovers the query result, $|D_i'| = Count(D_i')$, where $D_i' = \{D_k | TID_k = \lambda_i, d_k \in [L_i, U_i]\}$, by decrypting it.

B. Security Model

In Edge-PPMRQ, we assume that all entities are honest-but-curious, i.e., each entity performs the operations according to

the protocol, but they also want to reveal the privacy of other entities. For example, IoT devices and the edge server are curious about the query ranges $[L_i, U_i]$ and the corresponding query results $|D'_i|$, where $1 \leq i \leq m$; the edge server and the query user are also curious about the sensed data d_k of IoT device D_k. This paper focuses on the challenge of the privacy-preserving multi-dimensional range query. Similar to the preview work, we assume that there is no collusion among the entities. Meanwhile, we don't consider active attacks of an external adversary, which will be discussed in future work.

C. Design Goals

Based on the aforementioned system and security models, a privacy-preserving multi-dimensional range query scheme for edge-supported IIoT should achieve the following goals.

1) *Multi-dimensional range query*: Most of traditional range query schemes only support the single-dimensional range query in a query request, while the multi-dimensional range query scheme can realize the query for multi-dimensional data by sending only one query request.
2) *Privacy-preserving*: The query ranges $[L_i, U_i]$ and the query result $|D'_i|,\ i \in [1, m]$, cannot be revealed by any other entities except the query user. Besides, the edge server and the query user not only cannot get IoT devices' plaintext of sensed data, but also cannot determine the if the sensed data of IoT devices is within the query ranges.
3) *Continuous, discontinuous and arbitrary boundary range query*: Most of the traditional range query schemes only support continuous range query, while in reality, the query ranges may be discontinuous and arbitrary boundary. Consequently, it is very necessary for the multi-dimensional range query schemes to support all the three types of range queries.
4) *Communication and computationally efficient*: Compared to achieving multi-dimensional range query via single-dimensional range query schemes, a multi-dimensional range query scheme should significantly improve the efficiency of communication and computation. Especially, in edge-supported IIoT, the communication and computation loads of IoT side should be minimized as much as possible.

IV. The Proposed Scheme: Edge-PPMRQ

In this section, we introduce a privacy-preserving multi-dimensional range query scheme for edge-supported IIoT (Edge-PPMRQ). Before the detailed description of the scheme, we first present our ranges division algorithm, which is a critical technology in Edge-PPMRQ. Besides, the notations used in Edge-PPMRQ are listed in Table I.

TABLE I: Notations

Notation	Description
N	The number of IoT devices
n	The maximum value of IoT devices' sensed data
d	The data example, which is a value between 1 and n
D_k, d_k	The k-th IoT device and its sensed data, $1 \leq k \leq N$
m	The number of query dimensions
t	The number of query sub-ranges
λ_i	The index of i-th dimension
λ	The set of λ_i
TID_i	The dimension identifier of D_i's sensed data
TID	The set of TID_i
$[L_i, U_i]$	The query range for i-th dimension data
$[l_j, u_j]$	The j-th query sub-range
D'_i	The set of IoT devices, whose sensed data is in $[L_i, U_i]$
BF_j	The j-th bloom filter
BF	The set of bloom filters
P_f	The false positive rate of BF
$E_{ij}(r)$	The cipher of λ_i's tab corresponding to BF_j, $r \in \{0, 1\}$
EM	The set of $E_{ij}(r)$
C_{ij}	The λ_i's counter corresponding to BF_j
C	The set of C_{ij}
κ	The security parameter to establish OU encryption
pk, sk	The public key and private key of OU encryption
s	The number of elements mapped into bloom filter
s_j	The length of sub-range $[l_j, u_j]$
p	A large prime number
g	A primitive root of Z_p^*
a, g^a	The query user's private and public parameters
b, g^b	The IoT devices' private and public parameters
g^{ab}	The key shared among query user and IoT devices
h	A public hash function
h_k	The keyed hash value of d_k, i.e., $h(g^{ab} \| d_k)$

A. Ranges division algorithm

In order to achieve the multi-dimensional range query, multi-dimensional query ranges should be processed by the ranges division algorithm as the following steps.

1) *Query bounds extraction:* Suppose that $[L_1, U_1]$, $[L_2, U_2]$, ..., and $[L_m, U_m]$ represent the query ranges of m dimensions $\lambda_1, \lambda_2, ...,$ and λ_m, respectively. For $\forall i \in [1, m]$, we have $1 \leq L_i \leq U_i \leq n$. Then, all the lower bounds and upper bounds are extracted to a set $Q_1 = \{L_1, U_1, L_2, U_2, ..., L_m, U_m\}$. After that, both the minimum value 1 and maximum value n are inserted into Q_1. Finally, $Q_1 = \{1, L_1, U_1, L_2, U_2, ..., L_m, U_m, n\}$.
2) *Query bounds sort*: In order to make the ranges division easier, the set Q_1 is sorted from smallest to largest and the repeated bounds are deleted. Finally, a sorted set $Q_2 = \{l_1, l_2, ..., l_t, l_{t+1}\}$ is generated, where $l_1 = 1, l_{t+1} = n, l_j < l_{j+1}$ and $j \in [1, t]$.
3) *Query sub-ranges generation:* According to Q_2, t sub-ranges are generated, i.e., $[l_1, l_2]$, $[l_2, l_3]$, ..., and $[l_t, l_{t+1}]$. For clarity, we denote the sub-ranges as $[l_1, u_1]$, $[l_2, u_2]$, ..., and $[l_t, u_t]$. Furthermore, the bounds should be adjusted by adding 1, subtracting 1 or remaining unchanged to get final sub-ranges $[l'_1, u'_1]$, $[l'_2, u'_2]$, ..., and $[l'_t, u'_t]$, and the following two conditions should be satisfied for any query range $[L_i, U_i]$, $1 \leq i \leq m$:
   a) $\exists a, b, 1 \leq a \leq b \leq t, [l'_a, u'_a] \cup [l'_{a+1}, u'_{a+1}] \cup ... \cup [l'_b, u'_b] = [L_i, U_i]$, where $l'_a = L_i$ and $u'_b = U_i$.
   b) $\forall v, w \in [a, b]$, $v \neq w$, $[l'_v, u'_v] \cap [l'_w, u'_w] = \emptyset$.

To better understand our proposed ranges division algorithm, we give a toy example here, which is also shown in Fig. 2. For $n = 100, m = 2$ and two dimensions λ_1 and λ_2 with the corresponding query ranges $[20, 60]$ and $[40, 80]$, respectively, the query sub-ranges can be generated by the above ranges division algorithm:

1) According to $[20, 60]$ and $[40, 80]$, we can get $Q_1 = \{1, 20, 60, 40, 80, 100\}$.

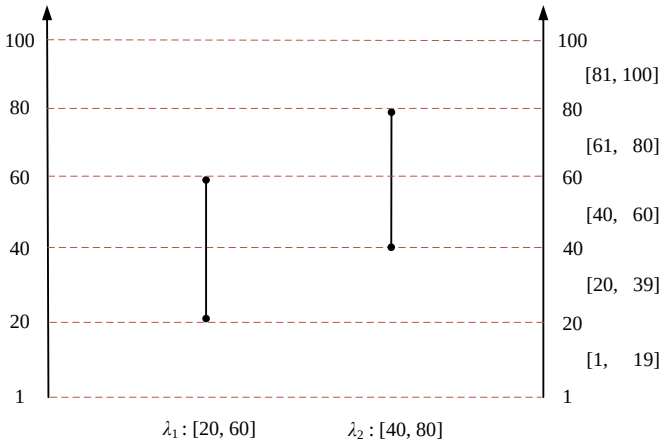


Fig. 2: An example of ranges division algorithm

2) Based on Q_1, we get a sorted set $Q_2 = \{1, 20, 40, 60, 80, 100\}$.
3) Then we can get 5 sub-ranges $[1, 20], [20, 40], [40, 60], [60, 80]$ and $[80, 100]$. Moreover, the bounds of the sub-ranges are adjusted to generate the final sub-ranges $[1, 19], [20, 39], [40, 60], [61, 80]$ and $[81, 100]$.
From the sub-ranges, we can see that:
   a) $[20, 39] \cup [40, 60] = [20, 60]$ and $[40, 60] \cup [61, 80] = [40, 80]$.
   b) $[20, 39] \cap [40, 60] = \emptyset$ and $[40, 60] \cap [61, 80] = \emptyset$.

B. Description of Edge-PPMRQ

We now describe our privacy-preserving multi-dimensional range query scheme for edge-supported IIoT (Edge-PPMRQ) in detail by using bloom filter data structure, OU encryption and ranges division algorithm. The five phases of Edge-PPMRQ are illustrated as below.

1) System initialization: To initialize the system, the service provider selects a hash function h, a large prime number p, a primitive root $g \in Z_p^*$ and a random number $b \in Z_p^*$. Then, g^b mod p can be computed accordingly. Finally, the service provider publishes the parameters h, p, g and g^b. After that, the service provider deploys an edge server and various dimensions IoT devices to the target industrial environment.

2) User query request generation: The query user launches a query for data of m dimensions $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$, i.e., "How many IoT devices of dimension λ_i, whose sensed data d_k is within the range $[L_i, U_i]$, where $1 \le i \le m$ and $1 \le k \le N$?" In other words, the query user would like to figure out: For $\forall i \in [1, m]$ and $k \in [1, N]$, $|D_i'| = Count(D_i')$, where $D_i' = \{D_k | TID_k = \lambda_i, d_k \in [L_i, U_i]\}$. To generate the query request in a privacy-preserving way, the query user performs the following steps.

Step 1: Given a security parameter κ, OU key generation algorithm outputs public key pk and private key sk. Besides, a random number $a \in Z_p^*$ is chosen to compute g^a mod p and $g^{ab} = (g^b)^a$ mod p, according to the public parameters p, g and g^b.

Step 2: For m query ranges $[L_1, U_1], [L_2, U_2], ...,$ and $[L_m, U_m]$ corresponding to dimensions $\lambda_1, \lambda_2, ...,$ and λ_m, respectively, the ranges division algorithm is called to output t query sub-ranges $[l_1, u_1], [l_2, u_2], ...,$ and $[l_t, u_t]$.

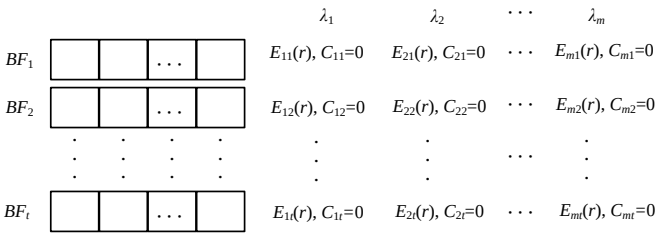


Fig. 3: Query request generation

Step 3: A group of t bloom filters with n-bit vectors and k hash functions are chosen. Here, for each bloom filter, s, i.e. the number of mapped elements, should satisfy $s \le \lceil \frac{n}{\log n} \rceil$ and k should meet $k = \frac{n}{s} \ln 2$ to ensure the false positive rate $P_f = (1 - (1 - \frac{1}{n})^{ks})^k \approx (1 - e^{-\frac{ks}{n}})^k = n^{-\ln 2}$.

Step 4: For each sub-range $[l_j, u_j]$, $1 < j < t$, all integers between l_j and u_j are mapped into BF_j. Note that, we don't map the integer value $d \in [l_j, u_j]$ itself but its keyed hash value $h(g^{ab} \| d)$ into BF_j. Finally, the sub-ranges $[l_1, u_1], [l_2, u_2], ...,$ and $[l_t, u_t]$ are mapped into t bloom filters $BF_1, BF_2, ...,$ and BF_t, respectively. For sub-range $[l_i, u_i]$ with length s_i, where $1 \le i \le t$, if $s_i > \lceil \frac{n}{\log n} \rceil$, we divide it into some shorter sub-ranges to ensure the false positive rate. For simplicity, we assume that the length s_i is smaller than $\lceil \frac{n}{\log n} \rceil$.

Step 5: For dimension λ_i, the query user constructs a query set $\{BF_j, E_{ij}(r), C_{ij}\}$, where $i \in [1, m]$, $j \in [1, t]$ and $r \in \{0, 1\}$. BF_j is the bloom filter which the j-th sub-range $[l_j, u_j]$ is mapped into. $E_{ij}(r)$ is the OU ciphertext of r, where r is a label to indicate the relationship between the i-th dimension λ_i and the j-th bloom filter BF_j. Concretely, if the j-th sub-range $[l_j, u_j]$ (mapped into BF_j) is a sub-set of the dimension λ_i's query range $[L_i, U_i]$, the r is labeled as 1 ($E_{ij}(r) = E(1)$); otherwise, r is set as 0 ($E_{ij}(r) = E(0)$). C_{ij} is a counter with initial value of 0, which is used to record how many IoT devices of dimension λ_i, whose sensed data is in the j-th sub-range $[l_j, u_j]$. Finally, we get:

$$EM = \begin{bmatrix} E_{11}(r) & E_{12}(r) & \cdots & E_{1t}(r) \\ E_{21}(r) & E_{22}(r) & \cdots & E_{2t}(r) \\ \vdots & \vdots & \ddots & \vdots \\ E_{m1}(r) & E_{m2}(r) & \cdots & E_{mt}(r) \end{bmatrix},$$

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1t} \\ C_{21} & C_{22} & \cdots & C_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m1} & C_{m2} & \cdots & C_{mt} \end{bmatrix}.$$

Step 6: As shown in Fig. 3, the query user constructs a query request $\{\lambda, BF, EM, C, g^a \text{ mod } p\}$, where $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ and $BF = \{BF_1, BF_2, ..., BF_t\}$. Then the query request is sent to the edge server. Accordingly, the edge server broadcasts $\{\lambda, g^a \text{ mod } p\}$ to all IoT devices.

Based on the example given in ranges division algorithm, we further show the concrete processes of the user query request generation algorithm:

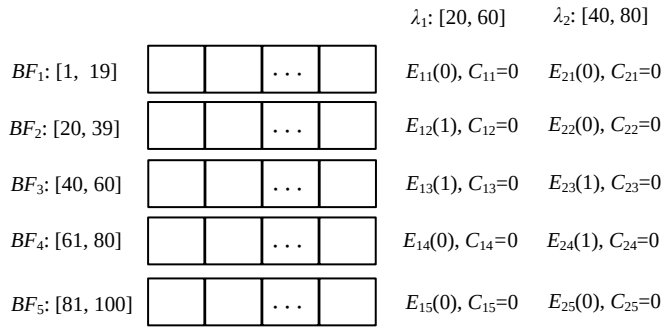


Fig. 4: An example of query request

1) For the 5 sub-ranges $[1,19]$, $[20,39]$, $[40,60]$, $[61,80]$ and $[81,100]$ generated by the ranges division algorithm, we map all the values in the sub-ranges into 5 bloom filters BF_1, BF_2, BF_3, BF_4 and BF_5, respectively. For instance, all the values $1, 2, ...,$ and 19 in the sub-range $[1,19]$ are mapped into BF_1.
2) For the query range $[20,60]$ of dimension λ_1, only $[20,39]$ and $[40,60]$ are its sub-sets. Therefore, only the tabs of $E_{12}(r)$ and $E_{13}(r)$ are 1. Then, the query user constructs a query set $\{BF_j, E_{1j}(r), C_{1j}\}$, $j \in [1,5]$ and $r \in \{0,1\}$, i.e., $\{\{BF_1, BF_2, BF_3, BF_4, BF_5\}, \{E_{11}(0), E_{12}(1), E_{13}(1), E_{14}(0), E_{15}(0)\}, \{C_{11}, C_{12}, C_{13}, C_{14}, C_{15}\}\}$. Similarly, for the query range $[40,80]$ of dimension λ_2, the query user can also generate the query set correspondingly. Finally, the query request, i.e., the two query sets, is generated as shown in Fig. 4.

3) IoT devices' data response: When receiving the data request $\{\lambda, g^a \bmod p\}$ forwarded by the edge server, each IoT device $D_k \in D$ checks if its type ID TID_k is in the queried dimensions $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$. If TID_k doesn't belong to λ, D_k discards the data request. Otherwise, D_k computes the shared key $g^{ab} \bmod p$ by using b and $g^a \bmod p$. Then, D_k computes the keyed hash value $h_k = h(g^{ab} \| d_k)$. Finally, $\{TID_k, h_k\}$ is responded to the edge server via a secure channel.

4) Edge server data aggregation: Upon receiving all the responses $\{TID_k, h_k\}, k \in [1, N]$ from IoT devices, the edge server performs the following steps to aggregate the data.

Step 1: Edge server takes TID_k from $\{TID_k, h_k\}$ and find out the corresponding query dimension λ_i from λ, where $1 \le i \le m$.

Step 2: Edge server takes h_k from $\{TID_k, h_k\}$ and checks if h_k is in BF_j, where $BF_j \in BF$ and $j \in [1,t]$. If so, the counter C_{ij} increases by 1. Otherwise, C_{ij} keeps unchanged.

Step 3: After the above processes, the edge server aggregates the data as $C_i = \prod_{j=1}^{t}(E_{ij}(r)^{C_{ij}})$, where $1 \le i \le m$, and responses the results $C = \{C_1, C_2, ..., C_m\}$ to the query user.

Following the last example, suppose that the data responses from IoT devices are $\{\lambda_1, h(g^{ab}\|5)\}$, $\{\lambda_1, h(g^{ab}\|20)\}$, $\{\lambda_1, h(g^{ab}\|30)\}$, $\{\lambda_2, h(g^{ab}\|35)\}$, $\{\lambda_2, h(g^{ab}\|45)\}$, $\{\lambda_1, h(g^{ab}\|50)\}$, $\{\lambda_1, h(g^{ab}\|55)\}$, $\{\lambda_2, h(g^{ab}\|55)\}$, $\{\lambda_2, h(g^{ab}\|65)\}$, $\{\lambda_2, h(g^{ab}\|90)\}$. The edge server performs the steps as below to aggregate the sensed data.

Step 1: When receiving the data responses, the edge server sorts the dimension λ_1's sensed data $h(g^{ab}\|5), h(g^{ab}\|20)$, $h(g^{ab}\|30)$, $h(g^{ab}\|50)$ and $h(g^{ab}\|55)$, and λ_2's sensed data $h(g^{ab}\|35), h(g^{ab}\|45)$, $h(g^{ab}\|55), h(g^{ab}\|65)$ and $h(g^{ab}\|90)$.

Step 2: It checks the bloom filter which the sensed data belongs to and increases the corresponding counter, e.g., $h(g^{ab}\|5)$ of λ_1 is in BF_1, so the corresponding counter C_{11} increases by 1. Finally, the counters $C_{11} = 1, C_{12} = 2, C_{13} = 2, C_{14} = 0, C_{15} = 0, C_{21} = 0, C_{22} = 1, C_{23} = 2$, $C_{24} = 1$ and $C_{25} = 1$.

Step 3: According to $E_{11}(0), E_{12}(1), E_{13}(1), E_{14}(0), E_{15}(0)$, $E_{21}(0), E_{22}(1), E_{23}(1), E_{24}(0)$ and $E_{25}(0)$ and counters, the data is aggregated as:

$$\begin{aligned} C_1 &= \prod_{j=1}^{5}(E_{1j}(r)^{C_{1j}}) \\ &= E_{11}(0)^1 \cdot E_{12}(1)^2 \cdot E_{13}(1)^2 \cdot E_{14}(0)^1 \cdot E_{15}(0)^1 \\ &= E_{11}(0) \cdot E_{12}(2) \cdot E_{13}(2) \cdot E_{14}(0) \cdot E_{15}(0) \\ &= E(4) \\ C_2 &= \prod_{j=1}^{5}(E_{2j}(r)^{C_{2j}}) \\ &= E_{21}(0)^0 \cdot E_{22}(0)^1 \cdot E_{23}(1)^2 \cdot E_{24}(1)^1 \cdot E_{25}(0)^1 \\ &= E_{21}(0) \cdot E_{22}(0) \cdot E_{23}(2) \cdot E_{24}(1) \cdot E_{25}(0) \\ &= E(3) \end{aligned}$$

Finally, $C = \{C_1, C_2\}$ is replied to the query user.

5) User response recovery: When receiving $C = \{C_1, C_2, ..., C_m\}$ from the edge server, the query user recovers the corresponding query results $D' = \{|D'_1|, |D'_2|, ..., |D'_m|\}$ by decrypting C as

$$|D'_i| = Count(D'_i) = Dec(C_i), 1 \le i \le m$$

The correctness of the result is shown as

$$\begin{aligned} Dec(C_i) &= Dec(\prod_{j=1}^{t}(E_{ij}(r)^{C_{ij}})) \\ &= Dec(\prod_{BF_j \in [L_i, U_i]}(E_{ij}(1)^{C_{ij}}) \cdot \prod_{BF_j \notin [L_i, U_i]}(E_{ij}(0)^{C_{ij}})) \\ &= Dec(E(\sum_{BF_j \in [L_i, U_i]}(1 \cdot C_{ij}) + \sum_{BF_j \notin [L_i, U_i]}(0 \cdot C_{ij}))) \\ &= Dec(E(\sum_{BF_j \in [L_i, U_i]} C_{ij})) \\ &= \sum_{BF_j \in [L_i, U_i]} C_{ij} \\ &= |D'_i|. \end{aligned}$$

In our example, when receiving $C = \{C_1, C_2\}$, the query user can recover the query result as:

$$|D_1'| = Count(D_1') = Dec(C_1) = 4.$$
$$|D_2'| = Count(D_2') = Dec(C_2) = 3.$$

Therefore, the query user knows that there are 4 sensed data of dimension λ_1 in the query range $[20, 60]$ and 3 sensed data of dimension λ_2 in the range $[40, 80]$.

V. Security Analysis

In this section, we analyze the security features of Edge-PPMRQ and prove that it achieves the aforementioned security requirements. We especially focus on privacy-preserving properties as below.

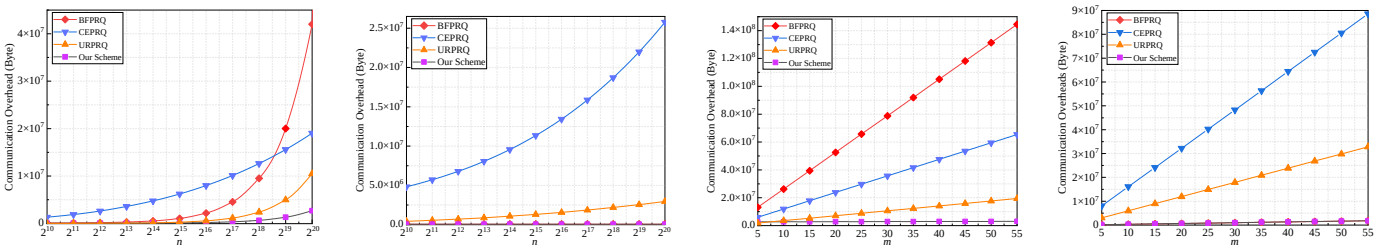
1) *Query ranges $[L_i, U_i], i \in [1, m]$ are privacy-preserving in Edge-PPMRQ*: In order to achieve multi-dimensional range query, the m-dimensional query ranges are divided into t sub-ranges by the proposed ranges division algorithm. Then the sub-ranges are mapped into t bloom filters to generate the query request. Subsequently, the query request is transferred to the edge server by the query user via a public channel. Through the channel, an IoT device D_k can eavesdrop the query request. At the same time, $D_k \in D$ owns shared key g^{ab} mod p. Therefore, for $\forall d \in [1, n]$, D_k can compute all keyed hash values $h(g^{ab}\|d)$ and check the bloom filter which $h(g^{ab}\|d)$ belongs to. In such way, D_k can recover the sub-range $[l_j, u_j]$, which is mapped into BF_j, $j \in [1, t]$. However, D_k cannot distinguish the ciphertexts of 0 and 1 because OU encryption is a semantically secure cryptosystem [25], i.e., it cannot know the real value of the tab r in $E_{ij}(r)$. As a result, D_k cannot confirm if the query sub-range $[l_j, u_j]$ is a sub-set of the query range $[L_i, U_i]$. Due to the same reason, even though D_k knows sub-range $[l_j, u_j]$ is mapped into BF_j, D_k still cannot get any information about the query range $[L_i, U_i]$ of dimension λ_i. Besides, the edge server not only cannot get shared key g^{ab} mod p, but also cannot distinguish the ciphertexts of 0 and 1. Therefore, it cannot get any information about $[l_j, u_j]$ and $[L_i, U_i]$. Based on above analysis, the query ranges $[L_i, U_i]$ of dimension λ_i, $i \in [1, m]$ are privacy-preserving for IoT devices and the edge server in Edge-PPMRQ.
2) *The query result $D' = \{|D_1'|, |D_2'|, ..., |D_m'|\}$ is also privacy-preserving in Edge-PPMRQ*: For the edge server, it receives the query request $\{\lambda, BF, EM, C, g^a \text{ mod } p\}$ from the query user and the data response $\{TID_k, h_k\}$, $k \in [1, N]$, from IoT devices. Even though it can know the bloom filter which the keyed hash value $h_k = h(g^{ab}\|d_k)$ belongs to, it cannot confirm if a sub-range $[l_j, u_j]$, which is mapped into the bloom filter BF_j, is the sub-set of the query range $[L_i, U_i]$ due to the fact that it cannot distinguish the ciphertexts of 0 and 1, according to the semantic security of OU cryptosystem [25]. Therefore, the edge server cannot confirm if the sensed data d_k of IoT device D_k is in the query range $[L_i, U_i]$ of dimension λ_i, $i \in [1, m]$, i.e., it has no idea about that how many IoT devices whose sensed data is in range $[L_i, U_i]$. In other words, the query result D' is privacy-preserving for the edge server. Besides, each IoT device $D_k \in D$ owns g^{ab} mod p and can compute all keyed hash values of data in $[1, n]$. However, every IoT device transfers its data response via a secure channel. As a result, D_k's sensed data d_k cannot be recovered by other IoT devices. At the same time, $E_{ij}(r)$ is also indistinguishable for IoT devices. Thus, IoT devices have no way to know the query result D'. Based on the above analysis, we prove that the query result D' is privacy-preserving for the edge server and IoT devices.
3) *Plaintext data d_k of IoT device D_k is also privacy-preserving in Edge-PPMRQ*: For the edge server, it can get g^a mod p from the query request. However, b is the private parameter of IoT devices. Therefore, the edge server cannot compute g^{ab} mod p. In other words, it cannot recover d_k from $h_k = h(g^{ab}\|d_k)$. Similar to some previous work, in our model, the IoT devices send their data response via a secure channel. As a result, D_k's sensed data d_k cannot be recovered by other IoT devices. Likewise, the query user also cannot recover the sensed data of IoT devices. Based on the above analysis, the plaintext data d_k of IoT device D_k is also privacy-preserving.

VI. Performance Evaluation

In this section, we evaluate the performance of Edge-PPMRQ and three related work, i.e., BFPRQ [21], CEPRQ [22] and URPRQ [23] from the aspects of communication overhead and computational costs. Then, we compare these schemes in detail. Here, it should be noted that Edge-PPMRQ and BFPRQ [21] are based on OU encryption [25] and Paillier encryption [26], respectively, while CEPRQ [22] and URPRQ [23] are based on their own constructed homomorphic encryption SHE. All the schemes are simulated with Python 3.8, Gmpy 2 and Math library. The experiments are carried out on an Intel(R) Core (TM) i5-7400 CPU @3.00GHz with Windows 10 system and 24GB RAM. To ensure the fairness, we keep the false positive rate of bloom filters in BFPRQ [21] and Edge-PPMRQ as $n^{-\ln 2}$. Besides, we simulate m-dimensional range query of BFPRQ [21], CEPRQ [22] and URPRQ [23] by performing m single-dimensional query requests since they don't support multi-dimensional range query. Furthermore, we evaluate the communication overhead and computational costs of the four schemes in two conditions: (1) n is varying from 2^{10} to 2^{20}, while m is fixed at 16; (2) m is varying from 5 to 55, while n is fixed at 2^{20}. In addition, the detailed parameters setting and notations used in the comparison are shown in Table II.

A. Communication overhead

In this section, we compare the communication overhead of BFPRQ [21], CEPRQ [22], URPRQ [23] and Edge-PPMRQ. The communication overhead between edge server and query user includes both the query request and query response.



(a) Between edge server and query user with varying n

(b) Between edge server and IoT devices with varying n

(c) Between edge server and query user with varying m

(d) Between edge server and IoT devices with varying m

Fig. 5: Communication overhead comparison

TABLE II: The parameters setting and notations

Parameter	Value
κ	$\kappa = 512, \|p\| = \|q\| = \kappa$
h	The public hash function SHA-256
k	The number of hash functions k= log$n\times$ln2
N	The number of single dimension's IoT devices. N=1000
$\|\lambda_i\|$	The length of dimension index with the length of 6 bits
m	The number of dimensions in the query request
b	log$n$
$\|E_{paillier}\|$	The length of paillier ciphertext
$\|E_{SHE}\|$	The length of SHE ciphertext
$\|E_{OU}\|$	The length of OU ciphertext
$\|H\|$	The length of SHA-256's output

Besides, the communication overhead between edge server and IoT devices consists of data request and data response.

For condition 1, we calculate the communication overhead of the four schemes between the edge server and the query user, and between the edge server and IoT devces, which are respectively shown in table III and table IV. In order to compare them more intuitively, Fig. 5(a) and Fig. 5(b) depict the communication overhead in condition 1. From the Fig. 5(a), we can see that the communication overhead between the edge server and the query user in Edge-PPMRQ keeps very low with n varying from 2^{10} to 2^{20}, while in BFPRQ [21], CEPRQ [22] and URPRQ [23], they grow rapidly. Fig. 5(b) shows the communication overhead comparison between edge server and IoT devices, from which we can know that both Edge-PPMRQ and BFPRQ [21] are equally efficient and perform significantly better than CEPRQ [22] and URPRQ [23].

For condition 2, we also analyze the communication overhead of the four schemes between the edge server and query user, and between the edge server and IoT devces, which are respectively shown in table V and table VI. Similarly, Fig. 5(c) and Fig. 5(d) intuitively present the communication overhead of the four schemes in condition 2. From the Fig. 5(c), the communication overhead between the edge server and the query user of the four schemes all show a linear growth trend, but the communication cost curve of Edge-PPMRQ is almost flat with varying m from 5 to 55, while the slopes of other schemes, BFPRQ [21], CEPRQ [22] and URPRQ [23], are much greater than that of Edge-PPMRQ. The reason is that Edge-PPMRQ achieves multi-dimensional range query by a query request, which saves a large number of communication cost for multi-dimension query range. Fig. 5(d) depicts the communication overhead between the edge server and IoT devices. From the figure, we find that the communication overhead in both Edge-PPMRQ and BFPRQ [21] keeps efficient, but the communication of CEPRQ [22] and URPRQ [23] are almost 8 times and 50 times of that in Edge-PPMRQ, respectively. This is because in CEPRQ [22] and URPRQ [23], IoT devices not only receive the query request forwarded by the edge server, but also send ciphertext response back, which result in a lot of communication overhead, while IoT devices of Edge-PPMRQ and BFPRQ [21] only send keyed hash values $h(g^{ab}\|d_k)$ and dimension index λ_i to the edge server, which are far shorter than ciphertext.

Based on the above comparisons, we can conclude that Edge-PPMRQ is remarkably communication-efficient.

B. *Computational cost*

In this section, we compare the execution time at the query user, IoT devices and the edge server with varying n and m, and the experiment comparison results are taken from the average value of 10 times simulations.

Fig. 6 shows the average computational time of BFPRQ [21], CEPRQ [22], URPRQ [23] and Edge-PPMRQ in condition 1. Specifically, Fig. 6(a) illustrates the execution time of the query user, including user query request generation and user response recovery. From the figure, we can see that Edge-PPMRQ is really efficient, while the execution time of BFPRQ [21], CEPRQ [22] and URPRQ [23] increases rapidly with n varying from 2^{10} to 2^{20}. For example, when $n = 2^{20}$, the execution time of BFPRQ [21] and CEPRQ [22] are 4 times and 7 times of that in Edge-PPMRQ, respectively. URPRQ [23] also obviously consumes more execution time than Edge-PPMRQ. Fig. 6(b) shows the execution time at IoT devices. The figure shows that in URPRQ [23], IoT devices undertake a lot of computational costs, while the execution time of other schemes, BFPRQ [21], CEPRQ [22] and Edge-PPMRQ, is negligible. Furthermore, a sub-figure in Fig. 6(b) is given to better illustrate the comparison among BFPRQ [21], CEPRQ [22] and Edge-PPMRQ. From the sub-figure, we can see that the execution time of CEPRQ [22] also grows more rapidly than Edge-PPMRQ and BFPRQ [21]. The reason is that in CEPRQ [22] and URPRQ [23], IoT devices perform a lot of time-consuming operations, i.e., homomorphic addition and homomorphic xor, while in Edge-PPMRQ and BFPRQ [21], IoT devices only compute the keyed hash value, which is

TABLE III: Communication overhead between the edge server and the query user with varying n

Schemes	Query request (bits)	Query result (bits)	Total overhead (bits)
BFPRQ	$m \cdot [b \cdot (n + \lvert E_{paillier} \rvert) + \lvert \lambda_i \rvert)]$	$m \cdot (\lvert E_{paillier} \rvert + \lvert \lambda_i \rvert)$	$16 \cdot b \cdot n + 32768 \cdot b + 32864$
CEPRQ	$m \cdot [(2 + \frac{2 \cdot b^3 + 3 \cdot b^2 - 11 \cdot b}{6}) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$\frac{2560}{3} \cdot b^4 + \frac{6400}{3} \cdot b^3 + \frac{10240}{3} \cdot b^2 + \frac{124160}{3} \cdot b + 46176$
URPRQ	$m \cdot [(b-1) \cdot (b+2) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$2560 \cdot b^3 + 2560 \cdot b^2 + 96$
Edge-PPMRQ	$b \cdot (n + m \cdot \lvert E_{OU} \rvert) + m \cdot \lvert \lambda_i \rvert$	$m \cdot (\lvert E_{OU} \rvert + \lvert \lambda_i \rvert)$	$b \cdot n + 24576 \times b + 24672$

TABLE IV: Communication overhead between the edge server and the IoT devices with varying n

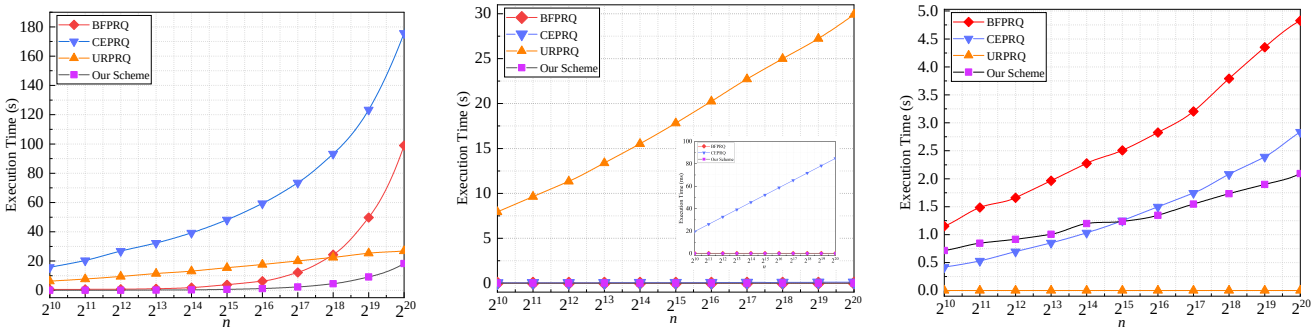
Schemes	Data request (bits)	Data response (bits)	Total overhead (bits)
BFPRQ	$m \cdot \lvert \lambda_i \rvert$	$m \cdot N \cdot (\lvert H \rvert + \lvert \lambda_i \rvert)$	4192096
CEPRQ	$m \cdot [(2 + \frac{2 \cdot b^3 + 3 \cdot b^2 - 11 \cdot b}{6}) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot N \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$\frac{2560}{3} \cdot b^4 + \frac{6400}{3} \cdot b^3 + \frac{10240}{3} \cdot b^2 + \frac{124160}{3} \cdot b + 46176$
URPRQ	$m \cdot [(b-1) \cdot (b+2) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot N \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$2560 \cdot b^3 + 2560 \cdot b^2 + 96$
Edge-PPMRQ	$m \cdot \lvert \lambda_i \rvert$	$m \cdot N \cdot (\lvert H \rvert + \lvert \lambda_i \rvert)$	4192096

TABLE V: Communication overhead between the edge server and the query user

Schemes	Query request (bits)	Query result (bits)	Total overhead (bits)
BFPRQ	$m \cdot [b \cdot (n + \lvert E_{paillier} \rvert) + \lvert \lambda_i \rvert)]$	$m \cdot (\lvert E_{paillier} \rvert + \lvert \lambda_i \rvert)$	$21014534 \cdot m$
CEPRQ	$m \cdot [(2 + \frac{2 \cdot b^3 + 3 \cdot b^2 - 11 \cdot b}{6}) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$9518886 \cdot m$
URPRQ	$m \cdot [(b-1) \cdot (b+2) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$858124 \cdot m$
Edge-PPMRQ	$b \cdot (n + m \cdot \lvert E_{OU} \rvert) + m \cdot \lvert \lambda_i \rvert$	$m \cdot (\lvert E_{OU} \rvert + \lvert \lambda_i \rvert)$	$30726 \cdot m + 20971520$

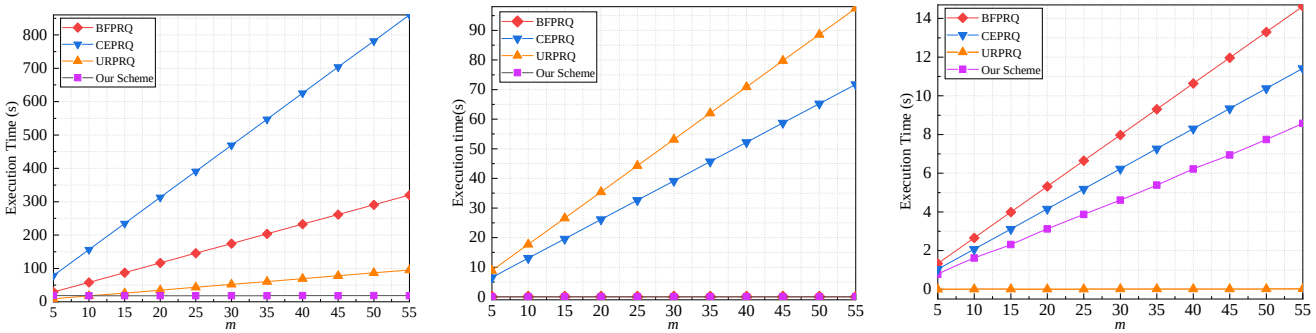
TABLE VI: Communication overhead between the edge server and the IoT devices

Schemes	Data request (bits)	Data response (bits)	Total overhead (bits)
BFPRQ	$m \cdot \lvert \lambda_i \rvert$	$m \cdot N \cdot (\lvert H \rvert + \lvert \lambda_i \rvert)$	$262000 \cdot m$
CEPRQ	$m \cdot [(2 + \frac{2 \cdot b^3 + 3 \cdot b^2 - 11 \cdot b}{6}) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot N \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$12881520 \cdot m$
URPRQ	$m \cdot [(b-1) \cdot (b+2) \cdot \lvert E_{SHE} \rvert + \lvert \lambda_i \rvert]$	$m \cdot N \cdot (\lvert E_{SHE} \rvert + \lvert \lambda_i \rvert)$	$2048000 \cdot m$
Edge-PPMRQ	$m \cdot \lvert \lambda_i \rvert$	$m \cdot N \cdot (\lvert H \rvert + \lvert \lambda_i \rvert)$	$262000 \cdot m$



(a) Time cost of user side (b) Time cost of IoT side (c) Time cost of edge server side

Fig. 6: Computational time cost comparison with varying n



(a) Time cost of user side (b) Time cost of IoT side (c) Time cost of edge server side

Fig. 7: Computational time cost comparison with varying m

much more efficient than homomorphic operations. Fig. 6(c) indicates the execution time at the edge server. From the figure, we can observe that Edge-PPMRQ, CEPRQ [22] and URPRQ [23] are more efficient than BFPRQ [21], and Edge-PPMRQ performs better than CEPRQ [22] when n is bigger than 2^{15}. Although the computational costs of the edge server in URPRQ [23] keep very low, it is at the cost of that a large amount of computational overhead for IoT devices according to Fig. 6(b).

Fig. 7 depicts the average computational time of BFPRQ [21], CEPRQ [22], URPRQ [23] and Edge-PPMRQ in condition 2. Specifically, Fig. 7(a) shows the execution time at the query user, including two parts, i.e., user query request generation and user response recovery. We can find that Edge-PPMRQ costs nearly constant execution time, which is apparently efficient than BFPRQ [21], CEPRQ [22] and URPRQ [23]. This is because the query user in Edge-PPMRQ performs less homomorphic operations than BFPRQ [21], CEPRQ [22] and URPRQ [23]. Fig. 7(b) depicts IoT devices' data response time, which shows that both Edge-PPMRQ and BFPRQ [21] are more efficient than CEPRQ [22] and URPRQ [23]. The reason is that IoT devices in Edge-PPMRQ and BFPRQ [21] only need compute keyed hash values $H(g^{ab}\|d_i)$, while both in CEPRQ [22] and URPRQ [23], IoT devices need perform a lot of homomorphic multiplication operations. Fig. 7(c) presents the data aggregation time cost at the edge server, and Edge-PPMRQ performs better than BFPRQ [21] and CEPRQ [22] since in Edge-PPMRQ, the number of homomorphic operations are less than that of BFPRQ [21] and CEPRQ [22]. Obviously, URPRQ [23] consumes the lowest computational time. The reason is that the most computational costs are afforded by IoT devices, and the edge server only performs cipher homomorphic addition operations.

C. Comprehensive comparison

Based on above introduce and analysis, a comprehensive comparison of Edge-PPMRQ and related schemes is given in table VII.

From a functional point of view, Edge-PPMRQ supports the functions of multi-dimension, discontinuous and arbitrary boundary range query, while BFPRQ [21], CEPRQ [22] and URPRQ [23] cannot support multi-dimension range query. Additionally, CEPRQ [22] also cannot support discontinuous ranges query and arbitrary boundary range query.

From a performance perspective, in edge/fog-supported IoT application, the most important mission of edge server/fog node is to reduce the burden of IoT devices, and the most tasks of IoT devices can be migrated to edge server/fog node. As a result, the communication overhead and computational costs of IoT devices can be greatly reduced, which not only remarkably extends the life period of IoT devices, but also achieves nearly real-time service. According to the comparison in the table VII, Edge-PPMRQ and BFPRQ [21] achieve the purpose well, while in CEPRQ [22] and URPRQ [23], IoT devices takes the most communication and computational tasks. Although BFPRQ [21] has low communication overhead between IoT devices and the edge server and low computational costs in IoT devices, its computational costs in edge server side is high. CEPRQ [22] has middle communication overhead between IoT devices and the edge server, and computational costs in the edge server, but it consumes high execution time in IoT devices. Moreover, URPRQ [23] not only has high communication overhead between IoT and edge server, but also consumes high computational costs in IoT devices. Although URPRQ [23] has lowest computational costs in fog node, it is at the cost of high computational load in IoT devices. Therefore, it is not suitable for edge/fog-supported IoT applications, in which the IoT devices own limited resource.

In a word, Edge-PPMRQ not only is functionally powerful for multi-dimension, discontinuous and arbitrary boundary range queries, but also achieves communication and computationally efficient for edge-supported IIoT application.

VII. Conclusion

Based on our proposed ranges division algorithm, this paper has designed a privacy-preserving multi-dimensional range query scheme for edge-supported IIoT. Edge-PPMRQ achieves the function of multi-dimensional range query, i.e. a user can query different types of data at once, which is very suitable for the real application of IIoT environment. Meanwhile, it also supports the continuous, discontinuous and arbitrary boundary range queries. The security analysis proves that Edge-PPMRQ is privacy-preserving, i.e. the query ranges and query results cannot be revealed by any other entities except the query user, and the sensed data of each IoT device cannot be recovered by other parties. Furthermore, a large number of experiments are conducted to evaluate the performance of Edge-PPMRQ and other related work, and the results show that Edge-PPMRQ is really communication and computationally efficient. Comprehensively, Edge-PPMRQ achieves expected goals in aspects of functions, privacy-preservation and efficiency. In future work, we plan to study more efficient range query schemes for various functions, privacy requirements and multiple application scenarios, e.g., multi-user range query scheme.

References

[1] D. Das, V. Hrishikesan, C. Kumar, and M. Liserre, "Smart transformer-enabled meshed hybrid distribution grid," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 1, pp. 282–292, 2021.

[2] A. Mackey, P. Spachos, and K. N. Plataniotis, "Smart parking system based on bluetooth low energy beacons with particle filtering," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3371–3382, 2020.

[3] A. R. Javed, L. G. Fahad, A. A. Farhan, S. Abbas, G. Srivastava, R. M. Parizi, and M. S. Khan, "Automated cognitive health assessment in smart homes using machine learning," *Sustainable Cities and Society*, vol. 65, p. 102572, 2021.

[4] J. Dong, M. Noreikis, Y. Xiao, and A. Ylä-Jääski, "Vinav: A vision-based indoor navigation system for smartphones," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1461–1475, 2019.

[5] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[7] X. Du, H. Chen, L. Zhu, J. Li, and Z. Chang, "Security and privacy in wireless iot," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 10–11, 2018.

TABLE VII: Comprehensive comparison

Schemes	Edge-PPMRQ	BFPRQ [21]	CEPRQ [22]	URPRQ [23]
Multi-dimension	√	×	×	×
Discotinuous range	√	√	×	√
Arbitrary boundary range	√	√	×	√
Communication overhead between IoT and edge server	low	low	middle	high
Computational costs in IoT	low	low	high	high
Computational costs in edge server	middle	high	middle	low

[8] Y. Yi, R. Li, F. Chen, A. X. Liu, and Y. Lin, "A digital watermarking approach to secure and precise range query processing in sensor networks," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 1950–1958.

[9] Y. T. Tsou, C. S. Lu, and S. Y. Kuo, "Ser: Secure and efficient retrieval for anonymous range query in wireless sensor networks," *Computer Communications*, vol. 108, pp. 1–16, 2017.

[10] J. Zeng, L. Dong, Y. Wu, H. Chen, C. Li, and S. Wang, "Privacy-preserving and multi-dimensional range query in two-tiered wireless sensor networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–7.

[11] L. Liu, Z. Hu, and L. Wang, "Energy-efficient and privacy-preserving spatial range aggregation query processing in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 7, 2019.

[12] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and scalable range query processing with strong privacy protection for cloud computing," *IEEE ACM Transactions on Networking*, vol. 24, no. 4, pp. 2305–2318, 2016.

[13] Z. Xu, Y. Lin, V. K. A. Sandor, Z. Huang, and X. Liu, "A lightweight privacy and integrity preserving range query scheme for mobile cloud computing," *Computers & Security*, vol. 84, pp. 318–333, 2019.

[14] X. Li, Y. Zhu, J. Wang, and J. Zhang, "Efficient and secure multi-dimensional geometric range query over encrypted data in cloud," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 44–54, 2019.

[15] J. Liang, Z. Qin, S. Xiao, J. Zhang, H. Yin, and K. Li, "Privacy-preserving range query over multi-source electronic health records in public clouds," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 127–139, 2020.

[16] P. Hu, Y. Wang, Q. Li, Y. Wang, Y. Li, R. Zhao, and H. Li, "Efficient location privacy-preserving range query scheme for vehicle sensing systems," *Journal of Systems Architecture*, vol. 106, p. 101714, 2020.

[17] J. Lim, K. Bok, and J. Yoo, "An efficient continuous range query processing scheme in mobile p2p networks," *The Journal of Supercomputing*, vol. 76, no. 2, pp. 1–15, 2020.

[18] X. Li, Z. Zhou, J. Guo, S. Wang, and J. Zhang, "Aggregated multi-attribute query processing in edge computing for industrial iot applications," *Computer Networks*, vol. 151, pp. 114–123, 2019.

[19] B. Djellabi, M. Younis, and M. Amad, "Effective peer-to-peer design for supporting range query in internet of things applications," *Computer Communications*, vol. 150, pp. 506–518, 2020.

[20] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K. K. R. Choo, "Multi-dimensional data indexing and range query processing via voronoi diagram for internet of things," *Future Generation Computer Systems*, vol. 91, pp. 382–391, 2018.

[21] H. Mahdikhani, R. Lu, Y. Zheng, and A. A. Ghorbani, "Achieving efficient and privacy-preserving range query in fog-enhanced iot with bloom filter," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[22] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5220–5232, 2020.

[23] H. Mahdikhani, R. Lu, J. Shao, and A. Ghorbani, "Using reduced paths to achieve efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4762–4774, 2020.

[24] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of The ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[25] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Advances in Cryptology-EUROCRYPT'98, International Conference on the Theory and Application of Cryptographic Techniques*, 1998, pp. 308–318.

[26] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology-EUROCRYPT'99, International Conference on the Theory and Application of Cryptographic Techniques*, 1999, pp. 223–238.