Computers, Materials & Continua
DOI: 10.32604/cmc.2022.023907
Article

Tech Science Press

Multi-dimensional Security Range Query for Industrial IoT

Abdallah Abdallah[1], Ayman A. Aly[2], Bassem F. Felemban[2], Imran Khan[3] and Ki-Il Kim[4,*]

[1]School of Engineering Technology, Al Hussein Technical University (HTU), Amman, 11831, Jordan
[2]Department of Mechanical Engineering, College of Engineering, Taif University, Taif, 21944, Saudi Arabia
[3]Department of Electrical Engineering, University of Engineering and Technology Peshawar, Pakistan
[4]Department of Computer Science and Engineering, Chungnam National University, Daejeon, 34134, Korea
*Corresponding Author: Ki-Il Kim. Email: kikim@cnu.ac.kr
Received: 26 September 2021; Accepted: 02 November 2021

Abstract: The Internet of Things (IoT) has allowed for significant advancements in applications not only in the home, business, and environment, but also in factory automation. Industrial Internet of Things (IIoT) brings all of the benefits of the IoT to industrial contexts, allowing for a wide range of applications ranging from remote sensing and actuation to decentralization and autonomy. The expansion of the IoT has been set by serious security threats and obstacles, and one of the most pressing security concerns is the secure exchange of IoT data and fine-grained access control. A privacy-preserving multi-dimensional secure query technique for fog-enhanced IIoT was proposed in light of the fact that most existing range query schemes for fog-enhanced IoT cannot provide both multi-dimensional query and privacy protection. The query matrix was then decomposed using auxiliary vectors, and the auxiliary vector was then processed using BGN homomorphic encryption to create a query trapdoor. Finally, the query trapdoor may be matched to its sensor data using the homomorphic computation used by an IoT device terminal. With the application of particular auxiliary vectors, the spatial complexity might be efficiently decreased. The homomorphic encryption property might ensure the security of sensor data and safeguard the privacy of the user's inquiry mode. The results of the experiments reveal that the computing and communication expenses are modest.

Keywords: Internet of things; data security; ciphertext; privacy encryption

1 Introduction

The Internet of Things (IoT) is regarded as the third revolution in the development of the information technology industry after computers and the Internet. Relying on the rapid development of smart homes, Internet of Vehicles (IoV), smart medical care, and smart grids based on the IoT, the era of the Internet of Everything (IoE) has arrived [1–5]. The latest statistics from the Statista website show that it is expected to exceed 30 billion interconnected devices in 2020 [6–10].

Countless smart terminals in the IoT transmit the massive amounts of data they sense through complex information and communication channels to data centers with huge data storage and processing capabilities, which can be abstracted as a "terminal transmission pipeline cloud" architecture, which also corresponds to the IoT three logical levels: perception layer, transmission layer and application layer [11–14]. Because of its ubiquitous network, comprehensive perception, reliable transmission, and intelligent processing, the characteristic elements are becoming more prominent, causing the IoT to face very severe security challenges [15].

The IIoT is the integration of the IoT system and the industrial automation system. It has the characteristics of comprehensive perception, interconnected transmission, and intelligent processing. It is one of the main development directions of the IoT technology in the future [16]. One of the important tasks of the IIoT application system is to analyze and process the sensor data of the IoT devices. If all the sensor data of the IoT devices are collected in the control center for processing, not only may the system be delayed due to network congestion, but also it will expose the system to the risk of a single point of failure [17–20]. The edge computing model that has emerged in recent years can effectively alleviate the performance bottlenecks and security risks that exist in the traditional centralized computing model by making full use of the computing power of the network edge devices [21–25]. By deploying fog devices at the edge of the IIoT, the sensor data in the IoT can be preprocessed on the edge of the network to improve the quality of service (QoS) [26–30].

The protection of sensor data of IoT devices is one of the important issues that need to be considered in fog-enhanced IIoT systems [31]. For example, in a fog-enhanced industrial Internet system, operation and maintenance personnel need to monitor whether the IoT devices in the system are operating normally, that is, query whether the sensor data of the IoT devices is in a reasonable range to determine their operating status. For security considerations, sensor data and query results cannot be leaked to any other unauthorized parties during the query process, so a secure data query plan must be designed [32]. In terms of data security query, many scholars have considered outsourcing the query of encrypted data. For example, the authors in [33] used Bloom filters to build a query index, which realized the fuzzy multi-keyword sort range query of encrypted data. The authors in [34] proposed a range query method that integrates bucket partitioning, identity authentication and verification code technologies for the data query problem in a two-layer wireless sensor network (WSN). The authors in [35] studied the multi-dimensional confidential range query of outsourcing data, and proposed an individualized query scheme with flexible permissions. In recent years, the range query for non-outsourced encrypted data has attracted the attention of researchers. For example, Reference [36] proposed a range query matrix method, and designed a single-dimensional security range query scheme using homomorphic encryption technology. Existing research on range query that supports privacy protection features mainly focuses on the query range and the confidentiality of the query subset that meets the conditions. Multi-dimensional query is not supported, and the communication overhead is relatively high.

Because some large-scale IoT devices in the IIoT have many different types of sensors, the monitoring of their status requires multiple sensor data belonging to the device, that is, multi-dimensional data query, while the traditional data query can only use multiple queries will increase the system's computing and communication costs. Therefore, it is necessary to design a multi-dimensional data query solution for the IIoT. In addition, the confidentiality of sensor data and the protection of user query mode need to be considered when designing the scheme. Constructing multi-dimensional data-oriented query trapdoors and matching based on homomorphic encryption is a feasible idea to solve this problem, such as the Boneh-Goh-Nissim (BGM) encryption algorithm [37] and the algorithm proposed by Reference [38]. The user performs confidential processing of the query range

to form a query trapdoor. After receiving the query trapdoor in the form of ciphertext, the IoT device matches its own sensor data, and uses the self-blindness of the homomorphic encryption algorithm to form a new sensor data and report the ciphertext to the fog device. The fog device collects the ciphertext of the sensor data that meets the conditions submitted by the IoT device to which it belongs, and returns the collected result to the inquiring user for calculation and decryption.

This paper proposes a multi-dimensional security range query solution with privacy protection features for the fog-enhanced IIoT. The solution uses query interval matrix generation, decomposition and reconstruction based on auxiliary vectors, and other methods to reduce the storage and communication costs. Then, uses the homomorphic encryption to achieve privacy protection. The proposed algorithm first maps the range of n-dimensional data involving multiple different dimensions and different starting points to a query matrix. The size of the matrix is determined by the total query interval size and is not affected by the required query dimension. Second, construct multiple auxiliary vectors to decompose and reconstruct the matrix. Third, we used the BGN homomorphic encryption to design the query trapdoor generation and matching. Finally, the feasibility and performance of the proposed and existing schemes are verified and analyzed through simulation experiments.

2 Pre-Knowledge

This section introduces the two basic mathematical methods used in the proposed algorithm, namely the large composite number $N = pq$-order bilinear mapping e: $G \times G \rightarrow G_T$ and the BGN homomorphic encryption algorithm.

2.1 Large Composite Order Bilinear Mapping

Let p, q be two large prime numbers of the same length, that is, their bit length $|p| = |q|$. Let $N = pq$, when there is a computable mapping relationship e that satisfies the following three properties between the group G and e: $G \times G \rightarrow G_T$, the mapping $e(G, G_T)$ can be called composite order bilinear mapping.

1) Bilinear. For any $(g, h) \in G^2$, $a, b \in \mathbb{Z}_N$ and $e(g^a, h^b) = e(g, h)^{ab}$.
2) Non-degenerate. There exists $g \in G$, so that $e(g, g)$ is on the group G_T of order N.
3) Computability. There is an efficient algorithm, when $(g, h) \in G$, all $e(g, h) \in G_T$ are computable.

The large composite order bilinear parameter generator $C\text{Gen}(K)$ is a probabilistic algorithm, which takes the safety parameter K as an input value and outputs a five-tuple (N, g, G, G_T, e). Among them, $N = pq$, p and q are prime numbers of 2 K bits. G and G_T are two groups of order N. $g \in G$ is a generator of order N of group G. e: $G \times G \rightarrow G_T$ is a non-degenerate, bilinear mapping that can be calculated efficiently.

Let g be a generator of G, then $h \in g^q \in G$ can generate a p-order subgroup $G_p = \{g^0, g^1, \ldots, g^{p-1}\}$, and $g' = g \in G$ can generate a q-order subgroup $G_q = G_p = \{g'^0, g'^1, \ldots, g'^{q-1}\}$. At this time, the problem of sub-group decision (SGD) on the group G [39] can be expressed as: given a five-tuple (N, g, G, G_T, e), if the element x is randomly from the group G or its subgroup G_q is selected, it is difficult to determine whether x is an element in G_q. If this problem is established, the security of the BGN homomorphic encryption algorithm is guaranteed [40–46].

2.2 BGN Homomorphic Encryption Algorithm

The BGN homomorphic encryption algorithm is a well-known fully homomorphic encryption algorithm, which consists of three phases, namely the key generation, encryption and decryption.

1) Key Generation Phase

Given the safety parameter K, the composite order bilinear mapping parameter group (N, g, G, G_T, e) is generated by the generator $C\text{Gen}(K)$. Here $N = pq$, where p, q are prime numbers of two K bits, G and G_T are two groups of order N, and $g \in G$ is a generator of order N of group G. Let $h = g^q$, h is a random generator of order p of G. At this time, the public key $pk = (N, G, G_T, e, g, h)$, and the secret key $sk = p$.

2) Encryption Phase

Suppose the message space which contains only integers and whose capacity is determined by the specific application $\hat{S} = \{0, 1, \ldots, \Delta\}$, $\Delta \ll q$. When encrypting $m \in \hat{S}$, select a random number $r \in \mathbb{Z}_N$, then the ciphertext $c = E(m, r) = g^m h^r \in G$.

3) Decryption Stage

Given the ciphertext $c = E(m, r) = g^m h^r \in G$, the plaintext can be recovered with sk. Observation shows that $c^p = (g^m h^r)^p = (g^p)^m$, if you want to decrypt m, it is equivalent to solving the discrete logarithm problem of c^p with g^p as the base, and since $0 \leq m \leq \Delta$, the time to solve this problem using the Pollard lambda algorithm and the complexity is $O(\Delta)$.

In addition, the BGN homomorphic encryption algorithm has self-blindness, that is, given the ciphertext $E(m, r) \in G$, $E(m, r + r') = E(m, r)h^{r'} \in G$ is valid ciphertext of m.

The BGN homomorphic encryption algorithm has the following homomorphic characteristics.

a) Additive homomorphism on group G. Given $E(m_1, r_1) \in G$ and $E(m_2, r_2) \in G$, we have

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 + m_2, r_1 + r_2) \in G \tag{1}$$

For the sake of simplicity, the random number term can be ignored and rewritten as $E(m_1)E(m_2) = E(m_1 + m_2) \in G$.

b) Multiplicative homomorphism on group G. Given $E(m_1, r_1) \in G$, and $m_2 \in \hat{S}$, there are $E(m_1, r_1)^{m_2} = E(m_1 m_2, r_1 m_2) \in G$, that is, $E(m_1)^{m_2} = E(m_1 m_2) \in G$.

c) Multiplicative homomorphism from group G to group G_T. Given $E(m_1) \in G$ and $E(m_2) \in G$, $e(E(m_1), E(m_2)) = E_T(m_1 m_2) \in G_T$.

d) Additive homomorphism on the group G_T. Given $E_T(m_1) \in G_T$ and $E_T(m_2) \in G_T$, $E_T(m_1)E_T(m_2) = E_T(m_1 + m_2) \in G_T$.

e) Multiplicative homomorphism on the group G_T. Given $E_T(m_1) \in G_T$, $m_2 \in \hat{S}$, $E(m_1)^{m_2} = E(m_1 m_2) \in G_T$.

3 Multi-Dimensional Query Scheme with Privacy Protection Features

The proposed algorithm is an aggregation query solution for various fog-enhanced IIoT. The multi-dimensionality in this article means that the sensor data of an IoT device is composed of data of multiple different dimensions. The data includes water temperature, voltage, volume, material allowance, etc. According to actual needs, it may be necessary to query the sensor data of different dimensions of the device at the same time. For example, in a factory, the average value of water temperature, voltage, volume, and material remaining can be counted, which cannot only understand and predict the operating status of the equipment in time, but also provide a basis for optimizing the production process.

It should be noted that in the actual IIoT environment, data have different dimensions and accuracy. For example, power consumption data, water consumption data, etc. may have decimals. Since each IoT device is unlikely to change its detection accuracy through software means such as over the air (OTA) upgrades after manufacturing, in order to simplify the calculations and adapt to the needs of multi-dimensional queries, the proposed algorithm is as follows. The sensing data with an accuracy of less than 1 needs to be converted when participating in the calculation. For example, the value of the power sensor data at a certain time node is 10.37 W. In the calculation, multiply it by 100, which becomes 1037 and then perform range query. In this way, the range query of multiple dimensions can be realized without loss of accuracy. In fact, the amplification factor of a certain dimension of data can be written into the device when it is deployed.

3.1 System Model

There are three types of entities in the proposed scheme, namely, fog equipment at the edge of the network, IIoT equipment $D = \{D_1, D_2, \ldots, D_N\}$ located within the jurisdiction of fog equipment, and query users. The proposed model is shown in Fig. 1.

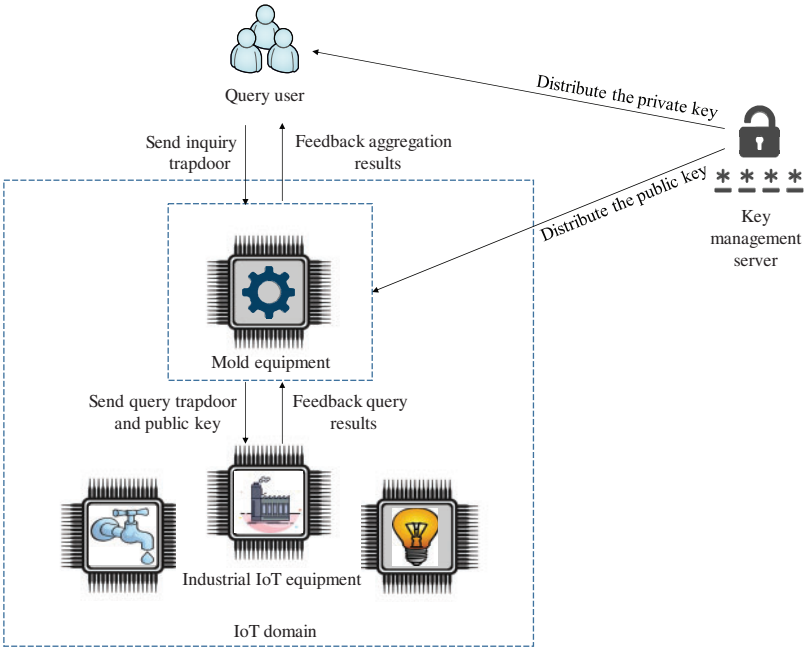

Figure 1: Proposed system model

1) Industrial Internet of Things equipment $D = \{D_1, D_2, \ldots, D_N\}$ is a collection of a group of devices, which are distributed in each specific domain. Each IoT device not only has the ability to detect and collect specific data, but also has the ability to transmit data, so that each IoT device D_k can periodically report sensor data to fog devices in its domain.
2) The fog equipment in the model of this paper is located at the edge of the network, and each fog equipment has its own jurisdiction. This jurisdiction can be determined according to specific application scenarios, such as a production equipment, a workshop, or even a factory area. The fog device can collect and calculate the sensor data of the device within its jurisdiction, and complete the query request sent by the user. Compared with IoT devices, fog devices have

stronger computing and storage performance, and can complete sensing data collection and response to IoT devices in real time.

3) In the model of the solution in this paper, the query user can directly generate multiple-dimensional range queries and send them to the fog device to obtain the required data from the fog device. For example, the user may want to know how many device sensor data within the range of fog devices can satisfy dimension 1, range $[B_1, T_1]$, dimension 2, range $[B_2, T_2]$, ..., dimension m, range $[B_m, T_m]$. The sensor data of which devices meet the conditions. What is the average value of sensor data in each dimension of the device that meets the conditions? The fog device can collect the sensory data that meets the conditions fed back from all devices within the jurisdiction, and then generate the correlation degree based on the feedback data, and finally return the correlation degree and sensor data to the user.

3.2 Multi-Interval Query Matrix

The proposed algorithm is improved on the basis of the single query interval matrix algorithm, so that the multi-dimensional range query can be matrixed while maintaining low communication overhead. Through observation, it can be found that any given query interval can be decomposed into two types of rows, namely partial rows (PR) and continuous rows (CR). Fig. 2 shows the query matrix after the mapping of two query intervals, the dark gray part is PR, and the light gray part is CR. This matrix structure provides the possibility to further compress the matrix. In general, a typical query interval can be decomposed to get two PR and one CR.

0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 2: The query matrix after the mapping of two query intervals

Suppose the total number of query intervals is n. First, define four special auxiliary vectors.

1) $\boldsymbol{X}_k = (x_{k1}, x_{k2}, \ldots, x_{kn})$. The generation rule is that when the i-th row elements in the k-th matrix are all 1, set $x_{ki} = 0$; otherwise, $x_{ki} = 1$.
2) $\underline{\boldsymbol{Y}}_k = (y_{k1}, y_{k2}, \ldots, y_{kn})$. The generation rule is that when there is at least one 1 in the j-th column of the k-th matrix, set $y_{kj} = 0$; otherwise, $y_{ki} = 1$.
3) $\boldsymbol{X}'_k = (x'_{k1}, x'_{k2}, \ldots, x'_{kn})$. The generation rule is, when the i-th row element in the k-th matrix contains at least one 1, set $x'_{ki} = 0$; otherwise, $x'_{ki} = 1$.
4) $\boldsymbol{Y}'_k = (y'_{k1}, y'_{k2}, \ldots, y'_{kn})$. The generation rule is, all $y'_{kj} = 1$.

This article deals with incomplete rows and continuous rows in the matrix respectively. First, split the incomplete rows corresponding to n query intervals into multiple matrices $\boldsymbol{R}_1, \mathbf{R}_2, \ldots, \boldsymbol{R}_{2n}$ independently, and split all consecutive rows into a matrix $\boldsymbol{R}_C$, as shown in Fig. 3. Obviously, the original query matrix $\boldsymbol{R} = \boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \ldots \cup \boldsymbol{R}_{2n} \cup \boldsymbol{R}_C$.

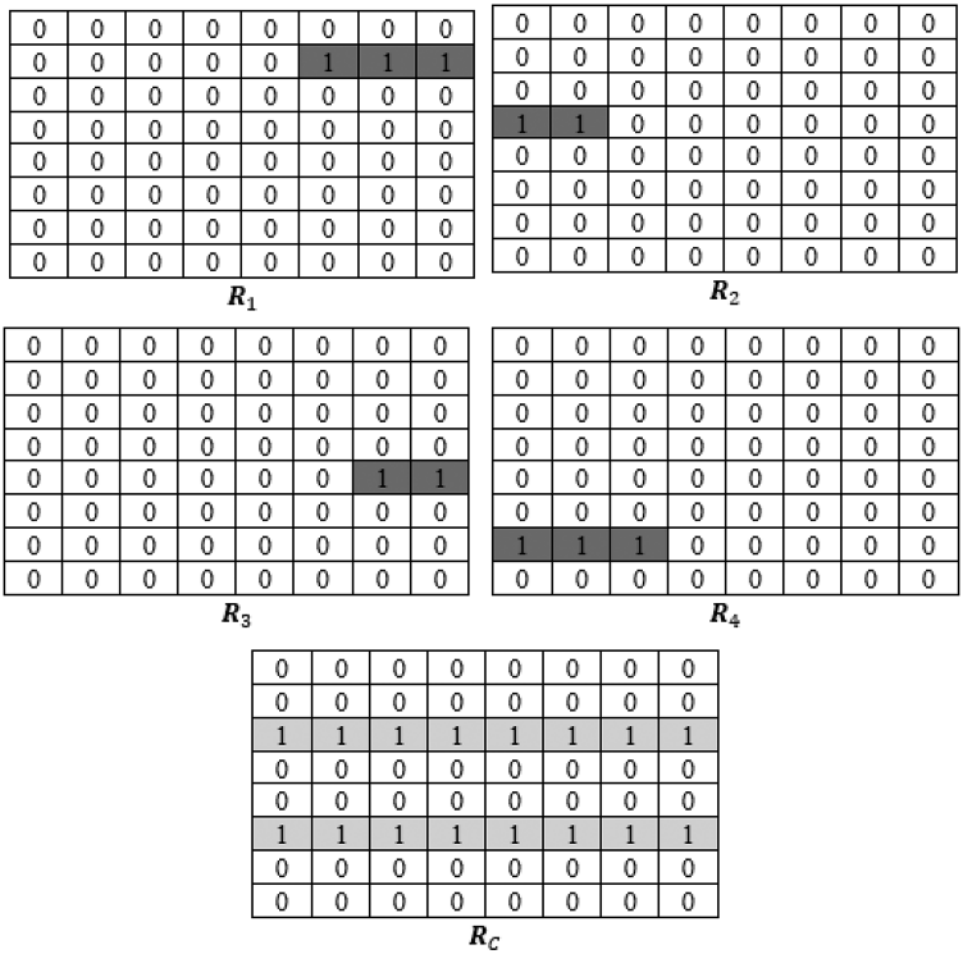

Figure 3: Query the split result of the matrix

Construct the vectors $\boldsymbol{X}_{\mathrm{k}}$, $\underline{\boldsymbol{Y}}_k$, $\boldsymbol{X}'_k$, $\boldsymbol{Y}'_k$ of all split matrices, and generate new matrices $1-\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{Y}_k$ and $\boldsymbol{X}_k^{'\mathrm{T}}\boldsymbol{Y}'_k$. According to the vector generation rules, each element in the two newly generated matrices can be expressed as

$$\begin{cases} R_{1-\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{Y}_k}(i,j) = 1 - x_{ki}y_{kj} \\ R_{\boldsymbol{X}_k'^{\mathrm{T}}\boldsymbol{Y}'_k}(_{i,j}) = x'_{ki}y'_{kj} = x'_{ki} \end{cases} \tag{2}$$

At this time, all elements in the matrix can be obtained through vector operations. Use $\boldsymbol{R}_k$ to represent the matrix related to incomplete rows, and use $\boldsymbol{R}_{\mathrm{C}}$ to represent the matrix related to continuous rows. The AND operation of the matrix is expressed as a bitwise AND operation, and the matrix OR operation is expressed as a bitwise OR operation and sorted into a vector multiplication.

The form of sum vector addition is shown in Eq. (3).

$$\begin{aligned} R(i,j) &= (\vee_{k=1}^{2n} R_k(i,j)) \vee R_C(i,j) \\ &= \vee_{k=1}^{2n} (R_{1-\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{Y}_k}(i,j) \wedge R_{\boldsymbol{X}_k'^{\mathrm{T}}\boldsymbol{Y}_k'}(_{i,j})) \\ &= \vee_{k=1}^{2n} ((1 - x_{ki}y_{kj}) \vee x'_{ki}y'_{kj}) \wedge ((1 - x_{Ci}y_{Cj}) \vee x'_{Ci}y'_{Cj}) \\ &= \sum_{k=1}^{2n} (1 - x_{ki}y_{kj})x'_{ki}y'_{kj} + (1 - x_{Ci}y_{Cj})x'_{Ci}y'_{Cj} \end{aligned} \tag{3}$$

Among the $8n+4$ vectors involved in the operation, there are $4n+1$ vectors whose elements are all 1, and donot need to be calculated and stored, namely $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{2n}, \boldsymbol{Y}'_1, \boldsymbol{Y}'_2, \ldots, \boldsymbol{Y}'_{2n}, \boldsymbol{Y}'_C\}$. Further, the elements in $\boldsymbol{Y}_C$ are all 0, and no storage and calculation are required. Therefore, only need to calculate and store $\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_{2n}, \boldsymbol{X}'_1, \boldsymbol{X}'_2, \ldots, \boldsymbol{X}'_{2n}, \boldsymbol{X}'_C, \boldsymbol{X}_C\}$ a total of $4n+2$ vectors to complete the calculation and matrix reconstruction. For any element $R(i,j)$ in the matrix, since $y'_{kj} = 1$, $x_{ki} = 1$ $(k = 1, 2, \ldots, 2n)$, and $y_{Cj} = 0$, $y'_{Cj} = 1$, its value can be calculated by Eq. (4).

$$R(i,j) = \sum_{k=1}^{2n} (1 - y_{kj})x_{ki} + x_{Ci} = \begin{cases} 1, & R(i,j) \text{ in the query interval} \\ 0, & \text{Other cases} \end{cases} \tag{4}$$

After calculation, it can be seen that the vector $\boldsymbol{X}_C$ has no effect in the reconstruction, and the final number of vectors required is $4n+1$.

3.3 Data Query Process

The program flow of the proposed method is mainly composed of three parts. First, the key management server generates the query key, distributes the private key to the user, and the public key to the fog device FD_i. When the user performs a data range query, the private key is used to encrypt the query range and query dimension to generate a trapdoor, and send it to the fog device FD_i. Then, after receiving the query trapdoor and query dimension ciphertext from the user, the fog device sends the query trapdoor to the IoT device under its jurisdiction for query. After the IoT device calculates the result ω of this query, it sends it to the fog device for aggregation. The fog device obtains ζ through aggregation and calculation and feeds it back to the user. Finally, the user calculates and decrypts the received ζ, and the query result can be obtained. The whole process is shown in Fig. 4. The following will specifically introduce the process of this article.

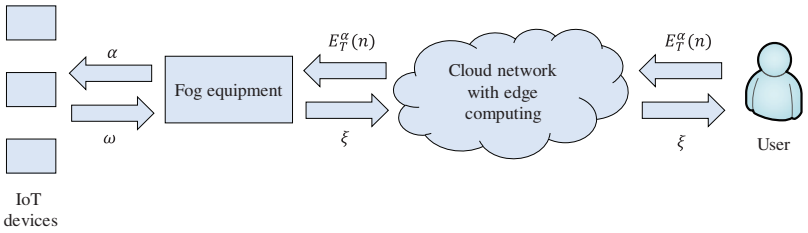

Figure 4: System flow

To facilitate the description of the data query process, the main parameters used in the process structure are listed in Tab. 1.

Table 1: Main parameters and their definitions

Parameter	Definition
σ	The matching degree of the query, the value is between 0 and the total number of dimensions
γ	The dimension of the query initiated by the user or the dimension of the IoT device itself
ω	The result of IoT device feedback to fog device
β	Query offset
B_i	Query the lower bound of the i-th interval
α	Query trapdoor
ξ	The query result sent by the fog device to the user
T_i	Query the upper bound of the i-th interval
Parse( )	Extract data from various types of composite data objects and vectors
Build( )	Generate various composite data objects and vectors, etc.

1) Key Generation

Given the security parameter K, the key management server generates a composite order bilinear mapping parameter set (N, g, G, G_T, e) by the generator $C\text{Gen}(K)$. Where $N = pq$, p, q are two K bit random prime numbers, G, G_T are two N-th order groups, $g \in G$ is an N-th order generator of group G, $e{:}G \times G \to G_T$ is large Composite order bilinear mapping. Set $h = g^q$, then h is a random generator of order p of G. The key management server generates public key $pk = (N, G, G_T, e, g, h)$, private key $sk = p$. After that, the key management server distributes the private key sk to the inquiring user, and distributes the public key pk to all fog devices FDi, and the fog device delivers to the IoT devices in its jurisdiction. The user sets its own message space as $\hat{S} = \{0, 1, \ldots, \Delta\}$, $\Delta \ll q$.

2) Query Trapdoor Generation

For each certain use scenario, the number and type of dimensions involved in each query initiated by the user are determined. In the proposed scheme, the number of dimensions of the data to be queried is set to n, that is, the number of intervals that users query is n. It should be noted that the query intervals $[B_{\text{query}}, T_{\text{query}}]$ referred to in the process are all multiplied sensing data intervals. The IoT has also performed the same multiplication processing on the data, so the data multiplication process will not be repeated. The trapdoor generation of a range query consists of the following steps.

a) Data Mapping and Conversion. Set any query dimension of this query as the first interval, and then determine the starting point of each interval in the query sequence according to the following rules. Among them, l_i is the interval length of the i-th interval.

$$
\begin{aligned}
&P_{1_{\text{start}}} = 0 \\
&P_{2_{\text{start}}} = l_1 + l_1 = 2l_1 \\
&P_{3_{\text{start}}} = 2l_1 + l_2 + l_2 = 2(l_1 + l_2) \\
&\vdots \qquad\qquad \vdots \\
&P_{n_{\text{start}}} = 2\sum_{i=1}^{n-1} l_i
\end{aligned} \tag{5}
$$

That is, the starting point of any i-th interval is the position after the end point of the $(i-1)$-th interval is increased by one interval length to prevent multiple intervals from being connected and forming too many CR blocks. After determining the starting point, the ending point of any i-th interval is

$$P_{i_{\text{end}}} = P_{i_{\text{start}}} + l_i = 2\sum_{t=1}^{i-1} l_t + l_i \tag{6}$$

For a query interval $[B_{\text{query}^i}, T_{\text{query}^i}]$, the k-th element u_{ik} in the query interval satisfies

$$u_{ik} = B_{\text{query}^i} + k \tag{7}$$

The data offset β_i of the query interval can be determined by the starting point of the query interval. After the offset, the k-th element in the query interval is v_{ik}, then β_i and v_{ik} satisfy the condition shown in Eq. (8).

$$\begin{aligned} &\beta_i = B_{\text{query}^i} - P_{i_{\text{start}}} \\ &v_{ik} = u_{ik} + \beta_i \end{aligned} \tag{8}$$

Determine the matrix order m from the end point of the last query interval, and then construct the m-order query matrix $\boldsymbol{R}$. In order to map the query interval to the query matrix $\boldsymbol{R}$, map v_{ik} to the matrix element $R(i, j)$, as shown in Eq. (9).

$$\begin{aligned} &m = \lceil P_{n_{\text{end}}} \rceil \\ &v_{ik} \to (i-1)m + j \\ &R(i-1) = \begin{cases} 1, & B_{\text{query}'} \le (i-1)m + j \le T_{\text{query}'} \\ 0, & other \end{cases} \end{aligned} \tag{9}$$

The order m of the matrix is determined by the end point of the last query interval rather than the upper bound of a single query interval, so the storage space requirement of each vector has nothing to do with the upper bound of a single interval of multi-dimensional query, only the length of all query intervals and related, the storage overhead is $O(m)$.

b) Vector Generation and Encryption. After generating the query matrix $\boldsymbol{R}$, according to the method described in Section 3.2, the corresponding vectors $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{X}'$, $\boldsymbol{Y}'$ are generated from the query matrix, and the required $4n+1$ vectors are selected for storage and encryption. which is

$$\begin{aligned} &2n \text{ for } \boldsymbol{Y}_k = (y_{k1},\ y_{k2}, \ldots, y_{km}) \\ &2n \text{ for } \boldsymbol{X}_k = (x'_{k1},\ x'_{k2}, \ldots, x'_{km}) \\ &\boldsymbol{X}'_C = (x'_{C1},\ x'_{C2}, \ldots, x'_{Cm}) \end{aligned} \tag{10}$$

For the convenience of calculation, the vector $\boldsymbol{Y}_k$ is used to further construct the vector $\bar{\boldsymbol{Y}}_k$, as shown in Eq. (11).

$$\bar{\boldsymbol{Y}}_k = (\bar{y}_{k1} = 1 - y_{k1}, \bar{y}_{k2} = 1 - y_{k2}, \ldots, \bar{y}_{km} = 1 - y_{km}) \tag{11}$$

According to Eq. (4), all elements $R(i,j)$ in the query matrix can be represented by the elements in the vector

$$R(i,j) = \sum_{k=1}^{2n}(1-y_{kj})x'_{ki} + x'_{Ci} = \sum_{k=1}^{2n}\bar{y}_{kj}x'_{ki} + x'_{Ci} \tag{12}$$

The user selects $2n+1$ random numbers $r_1, r_2, \ldots, r_{2n+1}$ to encrypt all $4n+1$ vectors to obtain

$$E(\bar{\boldsymbol{Y}}) = \{E(\bar{\boldsymbol{Y}}_1), E(\bar{\boldsymbol{Y}}_2), \ldots, E(\bar{\boldsymbol{Y}}_i)\},\ i = 1, 2, \ldots 2n$$

$$E(\boldsymbol{X}') = \{E(\boldsymbol{X}'_1), E(\boldsymbol{X}'_2), \ldots, E(\boldsymbol{X}'_i)\},\ i = 1, 2, \ldots 2n$$

$$E(\boldsymbol{X}'_C) = (g^{x'_{C1}}h^{r_{2n+1}}, g^{x'_{C2}}h^{r_{2n+1}}, \ldots, g^{x'_{Cm}}h^{r_{2n+1}})$$

where

$$E(\bar{\boldsymbol{Y}}_i) = \{E(\bar{\boldsymbol{Y}}_{i1}), E(\bar{\boldsymbol{Y}}_{i2}), \ldots, E(\bar{\boldsymbol{Y}}_{im})\} = \{g^{y'_{i1}}h^{r_i}, g^{y'_{i2}}h^{r_i}, \ldots, g^{y'_{im}}h^{r_i}\}$$

$$E(\boldsymbol{X}'_i) = \{E(\boldsymbol{X}'_{i1}), E(\boldsymbol{X}'_{i2}), \ldots, E(\boldsymbol{X}'_{im})\} = \{g^{x'_{i1}}h^{r_i}, g^{x'_{i2}}h^{r_i}, \ldots, g^{x'_{im}}h^{r_i}\} \tag{13}$$

In order to meet the needs of multi-dimensional query, two values need to be added to the original $E(\bar{\boldsymbol{Y}}_i)$, namely the queried dimension γ_i and the offset β_i required by the vector during operation.

$$E'(\bar{\boldsymbol{Y}}_i) = (\gamma_i, \beta_i, E(\bar{\boldsymbol{Y}}_{i1}), E(\bar{\boldsymbol{Y}}_{i2}), \ldots, E(\bar{\boldsymbol{Y}}_{im})) \tag{14}$$

Calculate the hash value $\boldsymbol{H}_i$ of the processed vector.

$$H = \{H_1, H_2, \ldots, H_i\}$$

$$H_i = \text{Hash}(E'(\bar{\boldsymbol{Y}}_i)||E(\boldsymbol{X}'_i)||E(\boldsymbol{X}'_C))$$

$$E'(\bar{\boldsymbol{Y}}) = \{E'(\bar{\boldsymbol{Y}}_1), E'(\bar{\boldsymbol{Y}}_2), \ldots, E'(\bar{\boldsymbol{Y}}_i)\}$$

$$E(\boldsymbol{X}') = \{E(\boldsymbol{X}'_1), E(\boldsymbol{X}'_2), \ldots, E(\boldsymbol{X}'_i)\}$$

$$\alpha = \{E'(\bar{\boldsymbol{Y}}), E(\boldsymbol{X}'), E(\boldsymbol{X}'_C), H\} \tag{15}$$

At this point, the query trapdoor α is generated. Send it to the fog device FD_i, and then send it to each IoT device D_k in its domain for query. In addition, the user also needs to generate and send the dimensional ciphertext $E_T(n)$ required for this query to the fog device FD_i.

$$E_T(n) = e(E(n), E(1)) \tag{16}$$

3) Processing on the Device Side of the IoT

Corresponding to the dimension identifier γ sent by the user terminal, each IoT device D_k has its own dimension identifier γ'_k. The IoT device sequentially extracts $E'(\bar{\boldsymbol{Y}}_i)$ in the query trapdoor α sent by the user and its corresponding hash value H_i for calculation and comparison. First, the IoT device calculates and compares the H_i sent by the user. If they are consistent, the dimension identifier γ_i in this vector is taken from $E'(\bar{\boldsymbol{Y}}_i)$ and compared with the dimension identifier γ'_k of the device itself. After comparison, the IoT devices screen out queries that meet their own dimensions, and extract related vectors from the query trapdoor together and form a **queryVector** for the next step of calculation. The specific process is shown in Algorithm 1. The time complexity of Algorithm 1 is $O(n)$ and is related to the number n of $E(\bar{\boldsymbol{Y}}_i)$ vectors sent by the user to query the trapdoor.

Algorithm 1: Equipment Comparison

Input: γ'_k, α
Output: queryVector
1: function DeviceCompare($E'(\bar{\boldsymbol{Y}})$, H)
2: $I \leftarrow 0$
3: **queryVector** $\leftarrow 0$
4: while $i = 1$ to n
5: Do
$H'_I \leftarrow \text{Hash}(E(\bar{\boldsymbol{Y}}_i)||E(\boldsymbol{X'}_i)||E(\boldsymbol{X'}_C))$
6: if $H'_I = H_i$
7: if $\gamma'_k = \gamma_i$
8: **queryVector** $\leftarrow$ Build($E'(\bar{\boldsymbol{Y}}_i)$, $E(\boldsymbol{X'}_i), E(\boldsymbol{X'}_{i+1}), E(\boldsymbol{Y'}_{i+1}), E'(\boldsymbol{X'}_C)$)
9: break
10: end if
11: end if
12: $i++$
13: end while
14: return **queryVector**
15: end function

After obtaining the **queryVector**, the IoT device first extracts the offset β_k of this query from it to offset its own sensor data v^*_k, obtain the offset value v'_k, and use it. The ElementShift function gets its position (i, j) in the matrix. At this time, the IoT device extracts the corresponding vector from the **queryVector** according to (i, j) for calculation. The specific process is shown in Algorithm 2. Each IoT device only needs to execute once, and the time complexity is $O(1)$.

Algorithm 2: IoT device handling

Input: γ'_k, m_k, **queryVector**
Output: ω_k
1: function DataShift(**queryVector**, m_k)
2: $\beta_k \leftarrow$ Parse(**queryVector**)
3: $v'_k \leftarrow v^*_k - \beta_k$
4: end function
5: function ElementShift(v'_k)
6: while $i = 1$ to m do
7: while $j = 1$ to m do
8: if $(i-1)m + j = v'_k$
9: break
10: end if
11: end while
12: end while
13: return i, j
14: end function
15: function DeviceCompute(v'_k, **queryVector**)
16: i, $j \leftarrow$ ElementShift(v'_k)

(Continued)

Algorithm 2: Continued

17: $E(\bar{\boldsymbol{Y}}_{1j}),\ E(\bar{\boldsymbol{Y}}_{2j}), E(x'_{1i}), E(x'_{2i}), E(x'_{Ci}) \leftarrow$ Parse(**queryVector**)
18: $r_{k1}, r_{k2} \leftarrow$ Random()
19: $c_k \leftarrow e(E(\bar{y}_{1j}), E(x'_{1i}))e(E(\bar{y}_{2j}), E(x'_{2i})), e(E(\bar{y}_{Cj}), E(x'_{Ci}))e(\mathrm{g}, h)^{r_{k1}}$
20: $s_k \leftarrow c_k^{v_k^*} e(\mathrm{g}, h)^{r_{k2}}$
21: $\omega_k \leftarrow \mathrm{Build}(c_k, s_k, \gamma'_k)$
22: return ω_k
23: end function

For the c_k and s_k described in Algorithm 2, there are

$$\begin{aligned} c_k &= e(E(\bar{y}_{1j}), E(x'_{1i}))e(E(\bar{y}_{2j}), E(x'_{2i})), e(E(\bar{y}_{Cj}), E(x'_{Ci}))e(\mathrm{g}, h)^{r_{k1}} \\ &= E_T(\bar{y}_{1j}x'_{1i} + \bar{y}_{2j}x'_{2i} + x'_{Ci}) = E_T(\boldsymbol{R}_k(i,j)) \end{aligned}$$

$$s_k = c_k^{v_k^*} e(\mathrm{g}, h)^{r_{k2}} = E_T(\boldsymbol{R}_k(i,j))^{v_k} e(\mathrm{g}, h)^{r_{k2}} = E_T(\boldsymbol{R}_k(i,j)v_k^*) \tag{17}$$

By executing Algorithms 1 and 2, the sensor data v_k^* of the IoT device can be converted into the mapping c_k of the corresponding position of the query matrix in the G_T group, so that a query is completed. In addition, the actual sensor data of the IoT device can also be returned through s_k in the case of query matching. After completing the calculation, send ω_k to the fog device FD_i of the domain to which the IoT device belongs.

4) Treatment of Fog Equipment

After the fog device FD_i receives the ω_k from the k subordinate IoT devices, it extracts the result c_k of the n-dimensional query by the IoT device from it and calculates it. The fog device multiplies all dimensional data c_k. According to the homomorphism of the BGN algorithm, the result obtained is the sum of all the results fed back on the k dimensions. The matching degree σ_i of this query of FD_i is sent to the user for this sum value. The difference of the encrypted query dimension information $E_T(n)$. The FD_i constructs σ_i and all the values of ω_k into ξ_i and sends it to the user. The specific process is shown in Algorithm 3. The time complexity $O(n)$ of Algorithm 3 is related to the number k of IoT devices to which the fog device FD_i belongs.

Algorithm 3: Fog Equipment Handling

Input: $\omega_1,\ \omega_2, \ldots,\ \omega_k, E_T(n)$
Output: ξ_i
1: function FDProcess($\omega_k, E_T(n)$)
2: while $k = 1$ to n do
3: $c_k \leftarrow \mathrm{Parse}(\omega_k)$
4: $\sigma_i \leftarrow \sigma_i c_k$
5: end while
6: $\sigma_i \leftarrow E_T(n)/\sigma_i$
7: $\xi_i \leftarrow \mathrm{Build}(\sigma_i, \omega_1, \omega_2, \ldots, \omega_k)$
8: return ξ_i
9: end function

5) User Analysis Data

After the user receives the data ξ_i sent by FD_i, first extract the query matching degree value σ_i and decrypt it. If and only if $\sigma_i = 0$, the result returned by the fog device FD_i completely matches the query. The user multiplies the completely matched data returned by the fog device by the sub-dimensions, and calculates the number of completely matched devices. The specific process is shown in Algorithm 4. The time complexity is related to the number n of fog equipment FD_i, which is $O(n)$.

Algorithm 4: User data analysis

Input: $\xi_1, \xi_2, \ldots, \xi_i$
Output: $S_1, S_2, \ldots, S_\gamma$, VD
1: function UserProcess($\xi_1, \xi_2, \ldots, \xi_i$)
2: while $i = 1$ to n do
3: $\sigma_i \leftarrow$ Parse(ξ_i)
4: $S_1, S_2, \ldots, S_\gamma \leftarrow 0$
5: VD $\leftarrow 0$
6: if Decrypt ($\sigma_i = 0$)
7: $\omega_1, \omega_2, \ldots, \omega_k \leftarrow$ Parse(ξ_i)
8: while 1 to k do
9: $s_k, \gamma'_k \leftarrow$ Parse(ω_k)
10: $S_{\gamma'_k} \leftarrow S_{\gamma'_k} s_k$
11: end while
12: VD++
13: else
14: $\xi_i \leftarrow$ NULL
15: end if
16: end while
17: return $S_1, S_2, \ldots, S_\gamma$, VD
18: end function

At this time, VD is the number of valid data returned. After decrypting S_γ, the sum of the data that meets the query conditions under the dimension γ can be obtained. After processing, the unmatched ξ_i are all set to 0, and the remaining elements come from the fog device that has returned valid data, and the specific fog device that has returned valid data can be located.

4 Safety Analysis

This section analyzes the security of the solution in this article, including forward security, backward security, privacy protection, unlinkability and other key security features, and introduces some attack modes designed for encryption range query solutions in recent years. The security of the proposed scheme is analyzed in the face of these attack modes.

4.1 Forward and Backward Safety

Assuming that the adversary A obtains the private key $sk = p$ used in a certain query, it can only decrypt the ciphertext of the current query. Since each time a query is made, the user will use the newly generated public and private key to query, the adversary A can only know the security parameter K used by the user to generate the key using the private key $sk = p$, and cannot be calculated from this.

Any information of the key previously generated by the user cannot decrypt any previously queried data except the current queried data. Therefore, the proposed scheme satisfies forward and backward security.

4.2 Privacy Protection in Query Mode

Suppose that adversary A intercepts the trapdoor used by user U in a certain query $\alpha = \{E(\bar{\boldsymbol{Y}}_i), E(X'), E(\boldsymbol{X'}_C), H\}$, because it can only be obtained in public key $\mathrm{pk} = (N, G, G_T, e, \mathrm{g}, h)$ and the encrypted information $E_{\mathrm{T}}(n)$ of the total dimension of the current query, so it cannot decrypt the specific content of the query trapdoor, and cannot know the current total number of query dimensions for each query. Furthermore, due to the semantic security of the BGN algorithm, the attacker cannot distinguish the content of the encrypted vector in the trapdoor, nor can it analyze the user's query rules by intercepting the trapdoor multiple times.

In addition to the adversary A, the IoT device D_k and the fog device FD_i also cannot know the specific information of the query trapdoor. Because the proposed scheme is different from the traditional query solution, as it uses homomorphic encryption calculation to replace the traditional comparison process. D_k determines its original sensor data v_k^* according to the query offset β and extracts the corresponding encryption in the query trapdoor for calculation. In this process, D_k cannot obtain the specific value of the query interval $[B_k, T_k]$. In the solution, FD_i only aggregates the feedback ω_k sent by the devices running in the IoT domain where it is located, and cannot know the specific query intervals in each dimension. Therefore, the query trapdoor in this paper will not reveal any useful information, such as query mode, query interval, etc., and the privacy of the query trapdoor is protected.

4.3 Sensing Data Confidentiality

Assume that adversary A intercepts the message ω_k sent by the IoT device D_k to the fog device FD_i in a certain query. Since it cannot obtain information other than the public key and the total number of dimensions in the current query in the public information, it cannot decrypt ω_k. It is impossible to know whether D_k satisfies this query or obtain the original sensor data v_k^* of D_k. Since the IoT device uses different random numbers when sending information to the fog device, the adversary A cannot analyze any useful information by intercepting ω_k multiple times.

Suppose that the adversary A intercepts the message ξ_i sent by the fog device FD_i to the user. Since it cannot obtain the private key $\mathrm{sk} = p$, it cannot decrypt ξ_i, and cannot obtain the IoT devices that meet the query conditions and their sensor data v_k^*. Same as the foregoing, the introduction of random numbers makes it impossible for adversaries to analyze and learn useful information such as query rules by intercepting ξ_i multiple times.

In addition to the adversary A, the fog device FD_i cannot learn the raw sensor data of the IoT device D_k v_k^* whether the query is satisfied. After FD_i receives $D_{k's}$ feedback ω_k, since it can only master the public key pk, it cannot decrypt the specific content of ω_k, and cannot know whether D_k meets this query or obtains the original sensor data v_k^* of D_k. Therefore, no party other than the inquiring user and the IoT device can obtain the sensor data v_k^* of the IoT device, and the confidentiality of the sensor data is guaranteed.

4.4 Unlinkability

In the proposed method, due to the use of random numbers, all query trapdoors generated by user U are random. Even if two or more query trapdoors are sent by the same user, external attackers

cannot determine whether these queries come from the same user, which means that the attacker cannot associate two different queries to a specific user. Therefore, the proposed solution provides unlinkability, and the attacker cannot associate the identity of the queryer by intercepting the query trapdoor.

4.5 Query Trapdoor Integrity

In order to prevent data integrity damage that may occur during data transmission, in the proposed solution, when user U generates a query trapdoor, paired query vectors $E(\bar{\boldsymbol{Y}}_i)$, $E(\boldsymbol{X}'_i)$ and $E(\boldsymbol{X}'_C)$ connect and calculate the message fingerprint Hash $(E(\bar{\boldsymbol{Y}}_i)||E(\boldsymbol{X}'_i)||E(\boldsymbol{X}'_C))$, and the generated hash value is sent to each IoT device D_k. Before extracting the content of a certain set of encrypted vectors, D_k first uses the hash value to verify the message. If the verification fails, the set of vectors is regarded as a damaged vector and discarded. In this way, the data integrity of the query trapdoor is guaranteed.

4.6 Key Update

In the proposed solution, the query process for the sensor data of IoT devices is essentially a BGN-based homomorphic encryption calculation process. The process does not involve decryption operations. The IoT devices and fog devices only need to know the system public key pk and the private key sk is only in the hands of the inquiring user. Therefore, there are fewer entities involved in key management and update, and the cost of key management is lower.

4.7 Protection Against Other Attack Methods

In addition to the above-mentioned common types of security attacks, in recent years, there have been a variety of attacks against encryption range queries. Among them, Reference [40] attacked encrypted data query by means of access mode disclosure. Reference [41] attacked the deterministic encryption (DTE) and order preserving encryption (OPE) methods in the attribute scheme, and used encrypted information and public auxiliary data to complete the plaintext recovery. Reference [42] defined two basic sources of leaks, such as access mode leaks and communication capacity leaks, and developed a universal one that is applicable to any system that supports range query and whose access mode or communication capacity is leaked. Reference [43] proposed an improved replay attack on encrypted data using the data leaked in the range query, including query mode and sorting information.

Different from the database query scenarios discussed in [40–43], the proposed scheme mainly considers the range query with privacy protection features in the fog-enhanced IIoT, which is an aggregate privacy protection query. The proposed scheme can protect the privacy of the upper and lower bounds in the range query and the subset of devices that meets the query conditions, and will not reveal the related information such as query mode and keyword frequency. In addition, for any range query, the data volume of the query results returned by the proposed scheme is the same, and no flow information will be leaked. Therefore, the proposed scheme can resist the above-mentioned several typical attacks on the security range query.

5 Performance Analysis

This section analyzes and compares the performance of the proposed scheme with the existing similar schemes, mainly including the comparison of the calculation and communication overheads of the query trapdoor generation phase and the server query phase. All simulation experiments were

carried out on a notebook computer configured with Intel i7-9750H 2.60 GHz, 16 GB memory, and running Windows 10 system, using Java Math's large number calculation library and JPBC library [44] to implement the algorithm. In addition, the operations that consume less time for addition operations, hash operations, etc., are ignored in the comparison.

5.1 Computational Overhead

5.1.1 Computational Overhead at Each Stage

For convenience, define T_m, T_o, T_p, and T_{em} to represent the time cost of single integer multiplication, integer modular power, bilinear mapping, and point multiplication on elliptic curves, respectively. Let l be the length of the query interval in each dimension, k is the total number of dimensions, and q is the upper bound of a single query interval. For solutions that only support single-dimensional queries, such as the Reference [36] solution, a multi-dimensional query is treated as multiple single-dimensional queries. Tab. 2 shows the comparison of the computational cost of each stage.

Table 2: Comparing the computational cost of each stage

Algorithm	Original data encryption stage	Trapdoor generation stage	Query phase
Reference [36]	-	$10k\lceil\sqrt{l}\rceil(T_o)$	$2kT_{\text{em}}+3kT_p+kT_o$
Reference [44]	$2klT_m+klT_o$	$4kT_m$	$2kT_m$
Reference [45]	$2klT_o+klT_m$	$(2k+1)T_o$	$3kT_m+2kT_o$
Reference [46]	$2klT_o+qT_{\text{em}}$	$5kT_m+6kT_o$	$2kT_{\text{em}}+kT_p$
Proposed	-	$(8k+2)\lceil\sqrt{l}\rceil(T_o)$	$2kT_{\text{em}}+3kT_p+kT_o$

Both the proposed scheme and Reference [36] method involve the IoT devices directly at the edge of the network to participate in the query, so the sensor data does not need to be encrypted and stored on the server, and the encryption phase does not generate computational overhead.

5.1.2 Full-Dimension Query Calculation Cost

This section considers the computational cost of querying data of all dimensions for each query when the total number of dimensions is certain. Suppose the total number of dimensions $k=10$, the length of the query interval $l=36$, and the upper bound of the query interval $q=50$. Then the References [36,44–46] algorithms and the proposed algorithm takes about 22.641, 15.864, 20.191, 21.638 and 20.360 ms to complete a full-dimensional query.

Fig. 5 shows the comparison result of the calculation cost of the full-dimension query of the specified dimensions for the scheme in this paper and the existing schemes. Among them, the proposed algorithm and Reference [36] schemes introduces the bilinear mapping operations in the query process, and the computational cost is higher than that in [44,45] that uses the integer encryption operations, but References [44–46] schemes need to encrypt and send the sensor data to a third-party data storage server in advance, this process will inevitably have time delay and generate encryption overhead. The scheme in [36] do not need to encrypt and send sensor data, and support users to query sensor data in real time. It only supports single-dimensional query, so the overall computational cost is higher than the propose scheme when multi-dimensional query is performed.

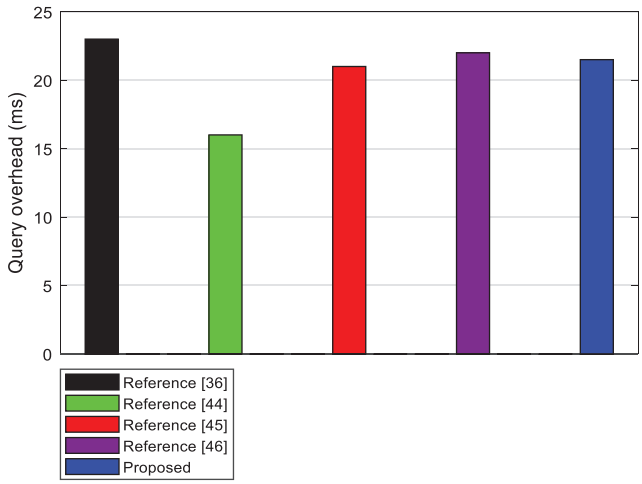

Figure 5: Comparing the calculation cost of full-dimensional query

5.1.3 Partial Dimensional Query Calculation Cost

Partial dimension quantity query means that when the total number of dimensions is certain, each query only queries the sensor data of part of the dimensions. Assuming that the total number of dimensions $k = 10$, the query interval length $l = 36$, and the upper bound of the query interval $q = 50$, part of the dimension queries with dimensions 5, 7, and 9 are performed respectively. Fig. 6 shows the comparison results of the calculation cost of the proposed scheme and the existing schemes when performing partial dimension query. It should be noted that, considering that even if part of the dimension query is performed, the amount of data stored by the data storage server should be the same as that of the full-dimensional query. Therefore, for a solution that needs to encrypt and store sensor data, the encryption overhead is still equivalent to full-dimensional query. Although the proposed algorithm and literature [36,46] all uses the bilinear mapping operation, the query stage overhead is high, but because the proposed and Reference [36] schemes are for real-time query of sensor data, they do not needs to be encrypted and stored in advance, so when fewer dimension queries are performed under the condition of a given total number of dimensions (5, 7 queries), the overall computing overhead is still lower than that of a solution that requires encrypted storage of sensor data. Since Reference [36] only supports single-dimensional query, it needs to repeatedly send query trapdoors when performing multi-dimensional query, and its overall computational cost is also higher than the proposed scheme.

5.1.4 Computational Overhead for Querying with Different Interval Lengths

The query of different interval length means that when the total number of dimensions, the upper limit of the query interval length, and the upper limit of the query interval are certain, each query only queries the finite interval length. Suppose the total number of dimensions $k = 10$, the upper limit of the query interval length $l = 100$, and the upper limit of the query interval $q = 100$. When querying 25%, 50%, and 75% length intervals respectively, the proposed algorithm and the query of different interval lengths refer to the case of the total number of dimensions, the upper limit of the query interval length, and the upper limit of the query interval, each query only queries the finite interval length. Suppose the total number of dimensions' $k = 10$, the upper limit of the query interval length $l = 100$, and the upper limit of the query interval $q = 100$. When querying 25%, 50%, and 75% length intervals respectively,

compare the proposed scheme with existing schemes when completing a query calculation cost, and the result is shown in Fig. 7.

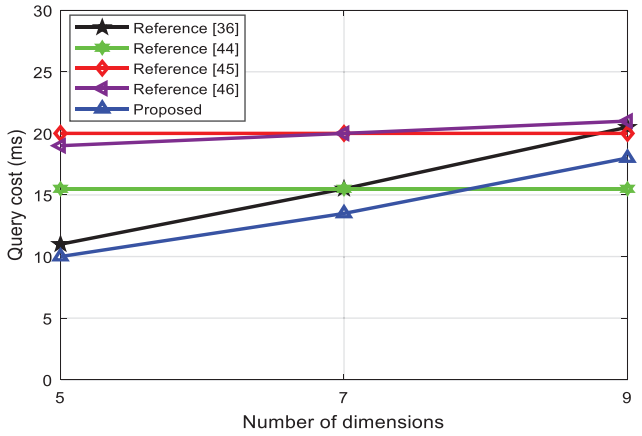

Figure 6: Comparing the calculation cost of different dimensions

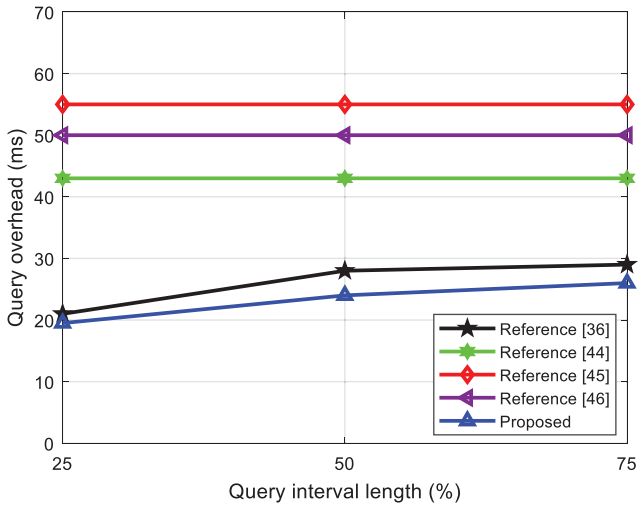

Figure 7: Comparing the computational cost of queries with different interval lengths

According to the analysis of the calculation cost of the query of the number of dimensions in this article, it can be seen that for the storage scheme that needs to encrypt the original sensor data, the encryption cost is the same as the calculation cost of the full-dimensional query. Except for the proposed and Reference [36] schemes, the computational cost of the other three schemes is not related to the length of the query interval, but is related to the total number of query dimensions. However, due to References [44–46] schemes, the original sensor data needs to be encrypted and stored in advance, so that when the query interval length is large and the total number of dimensions is high, the overall computational cost is relatively high. The Reference [36] scheme does not support multi-dimensional queries, and repeatedly sends single-dimensional query trapdoors when performing multi-dimensional queries. It uses more vectors than the proposed scheme, and the computational cost is higher.

5.2 Communication Overhead

This section compares the communication overhead of the proposed scheme and the existing schemes. Set the total number of dimensions $k = 10$, the length of the query interval $l = 36$, and the upper bound of the query interval $q = 50$ full-dimensional query, and compare the communication overhead of the whole process of different schemes. In order to compare the communication overhead of each scheme fairly, the following assumptions are made.

1) The security parameters (or key length) of each scheme are 128 bits.
2) The hash algorithm used in each scheme is SHA-1, therefore, the hash digest length of each scheme is 160 bits.
3) The random numbers used in each scheme are all 64 bits.

The communication process of the proposed scheme mainly lies in the sending stage of query trapdoor, the stage of fog device feedback from IoT devices and the stage of user feedback query from fog devices. Since the proposed scheme is to vectorize the query matrix, the communication overhead of the query trapdoor is relatively large, and the overhead of the other stages is relatively small. The communication cost of the query trapdoor generated of the proposed scheme during the sending phase of the query trapdoor is about 31016 B, the communication cost of the fog device phase of the IoT device feedback is about 1365 B, and the communication cost of the fog device feedback query user part is about 1370 B, a total of about 33751 B. Tab. 3 shows the communication overhead of the scheme in this article and the comparison scheme.

Table 3: Communication overhead comparison

Algorithm	Communication overhead (B)
Reference [36]	40535
Reference [44]	17520
Reference [45]	32880
Reference [46]	26830
Proposed	33751

Reference [44] scheme uses a homomorphic encryption algorithm in the integer domain improved from the SHE encryption scheme. Under the same conditions, the ciphertext length is small, but its algorithm is susceptible to noise in the ciphertext. When the noise in the ciphertext is large, it affects the decryption process. Reference [45] scheme has a slightly lower communication overhead than the proposed scheme, however this and Reference [44] schemes encrypt sensor data and query the data transmitted to the server, and it requires two servers to participate in the calculation, and the transmission delay is too large to realize the real-time query of sensor data. Reference [46] scheme query trapdoor is smaller in size, which makes its communication cost lower than the proposed and Reference [36] schemes, but its computational cost is high, and it does not support real-time query of sensor data. Reference [36] scheme that provides real-time data query only supports single-dimensional query, and multiple queries need to be initiated when querying a multi-dimensional sensor data, and its communication overhead is higher than the solution in this paper.

In summary, the proposed algorithm achieves a good balance in computing overhead, communication overhead, and support for multi-dimensional query features.

6 Conclusion

In this paper, a multi-dimensional security query scheme with privacy protection features with high communication efficiency is designed for the fog-enhanced industrial Internet of Things. The proposed scheme uses the BGN homomorphic encryption algorithm to encrypt the query trapdoor, and integrates matrix decomposition and reconstruction technology to protect the privacy of the query mode, while ensuring that the sensor data will not be leaked to unauthorized parties. Security analysis and simulation experiments show that the scheme in this paper can maintain a low level of computing and communication costs while ensuring multiple security features and multi-dimensional query functions. The next step will further optimize the proposed scheme, such as further improving communication efficiency and reducing communication overhead.

Acknowledgement: Taif University Researchers Supporting Project Number (TURSP-2020/260), Taif University, Taif, Saudi Arabia

Funding Statement: This study was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2019-0-01343, Training Key Talents in Industrial Convergence Security).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Sun, J. Liu, J. Wang, Y. Cao and N. Kato, "When machine learning meets privacy in 6G: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2694–2724, 2020.

[2] T. Huang, W. Yang, J. Wu, J. Ma, X. Zhang *et al.,* "A survey on green 6G network: Architecture and technologies," *IEEE Access*, vol. 7, pp. 175758–175768, 2019.

[3] A. A. Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] M. A. Garadi, A. Mohammed, A. K. Ali, X. Du, I. Ali *et al.,* "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[5] S. Li, L. Li, B. Xu, Y. Feng and H. Zhou, "Research of a reliable constraint algorithm on MIMO signal detection," *International Journal of Embedded Systems*, vol. 12, no. 2, pp. 13–26, 2020.

[6] J. Gojal, E. Monteiro and J. S. Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[7] K. Tange, M. D. Donno, X. Fafoutis and N. Dragoni, "A systematic survey of industrial internet of things security: Requirements and foq computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[8] S. Bashir, M. H. Alsharif, I. Khan, M. A. Albreem, A. Sali *et al.,* "MIMO-terahertz in 6G nano-communications: Channel modeling and analysis," *Computers, Materials & Continua*, vol. 66, no. 1, pp. 263–274, 2020.

[9] F. Jameel, T. Ristaniemi, I. Khan and B. M. Lee, "Simultaneous harvest-and-transmit ambient backscatter communications under Rayleigh fading," *EURASIP Journal on Wireless Communications and Networking*, vol. 19, no. 1, pp. 1–9, 2019.

[10] Q. Alsafasfeh, O. A. Saraereh, A. Ali, L. A. Tarawneh, I. Khan *et al.,* "Efficient power control framework for small-cell heterogeneous networks," *Sensors*, vol. 20, no. 5, pp. 1–14, 2020.

[11] K. M. Awan, M. Nadeem, A. S. Sadiq, A. Alghushami, I. Khan *et al.,* "Smart handoff technique for internet of vehicles communication using dynamic edge-backup node," *Electronics*, vol. 9, no. 3, pp. 1–17, 2020.

[12] W. Shahjehan, S. Bashir, S. L. Mohammed, A. B. Fakhri, A. A. Isaiah *et al.,* "Efficient modulation scheme for intermediate relay-aided IoT networks," *Applied Sciences*, vol. 10, no. 6, pp. 1–12, 2020.
[13] B. M. Lee, M. Patil, P. Hunt and I. Khan, "An easy network onboarding scheme for internet of things network," *IEEE Access*, vol. 7, pp. 8763–8772, 2018.
[14] O. A. Saraereh, A. Alsaraira, I. Khan and B. J. Choi, "A hybrid energy harvesting design for on-body internet-of-things (IoT) networks," *Sensors*, vol. 20, no. 2, pp. 1–14, 2020.
[15] T. Jabeen, Z. Ali, W. U. Khan, F. Jameel, I. Khan *et al.,* "Joint power allocation and link selection for multi-carrier buffer aided relay network," *Electronics*, vol. 8, no. 6, pp. 1–15, 2019.
[16] M. Waqas, Y. Niu, Y. Li, M. Ahmed, D. Jin *et al.,* "A comprehensive survey on mobility-aware D2D communications: Principles, practice and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1863–1886, 2020.
[17] M. Ahmed, Y. Li, M. Waqas, M. Sheraz, D. Jin *et al.,* "A survey on socially aware device-to-device communications," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2169–2197, 2018.
[18] F. S. Shaikh and R. Wismuller, "Routing in multi-hop cellular device-to-device (D2D) networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2622–2657, 2018.
[19] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma *et al.,* "Security and privacy in device-to-device (D2D) communication: A review," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1054–1079, 2017.
[20] O. Hayat, R. Ngah, S. Z. M. Hashim, M. H. Dahri, R. F. Malik *et al.,* "Device discovery in D2D communication: A survey," *IEEE Access*, vol. 7, pp. 131114–131134, 2019.
[21] A. Asadi, Q. Wang and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.
[22] B. M. Elhalawany, R. Ruby and K. Wu, "D2D communication for enabling internet-of-things: Outage probability," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2332–2345, 2019.
[23] I. Loannou, V. Vassiliou, C. Christophorou and A. Pitsillides, "Distributed artificial intelligence solution for D2D communication in 5G networks," *IEEE Systems Journal*, vol. 14, no. 3, pp. 4232–4241, 2020.
[24] P. Pawar, A. Trivedi and M. K. Mishra, "Outage and ASE analysis for power controlled D2D communication," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2269–2280, 2020.
[25] F. Qamar, M. U. A. Siddiqui, M. H. D. Hinidia, R. Hassan and Q. N. Nguyen, "Issues, challenges, and research trends in spectrum management: A comprehensive overview and new vision for designing 6G networks," *Electronics*, vol. 9, no. 9, pp. 1–124, 2019.
[26] Y. R. B. Al-Mayouf, N. F. Abdullah, O. A. Mahdi, S. Khan, M. Ismail *et al.,* "Real-time intersection-based segment aware routing algorithm for urban vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2125–2141, 2018.
[27] M. K. Hasan, M. M. Ahmad, A. H. A. Hashim, A. RAzzaque, S. Islam *et al.,* "A novel artificial intelligence based timing synchronization scheme for smart grid applications," *Wireless Personal Communications*, vol. 114, no. 2, pp. 1067–1087, 2019.
[28] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, A. K. Abasi *et al.,* "A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home," *Swarm and Evolutionary Computation*, vol. 60, no. 2, pp. 1–17, 2021.
[29] N. Nurelmadina, M. K. Hasan, I. Memon, R. A. Saeed, K. A. Z. Ariffin *et al.,* "A systematic review on cognitive radio in low power wide area network for industrial IoT applications," *Sustainability*, vol. 13, no. 1, pp. 1–21, 2021.
[30] K. R. Choo, R. X. Lu, L. Q. Chen and X. Yi, "A foggy research future: Advances and future opportunities in fog computing research," *Future Generation Computer Systems*, vol. 78, no. 2, pp. 677–679, 2018.
[31] R. X. Lu, K. Hueng, A. H. Lashkari and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.
[32] R. Lu, X. H. Liang, X. Li and X. Shen, "EPPA: An efficient and privacy-preserving aggregation scheme for smart grid communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1621–1631, 2012.

[33] J. Wang, X. Yu and M. Zhao, "Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query," *Arabian Journal for Science and Engineering*, vol. 40, no. 8, pp. 2375–2388, 2015.
[34] H. Dai, Q. Ye, X. Yi, R. He, G. Yang *et al.,* "VP2RQ: Efficient verifiable privacy-preserving range query processing in two-tiered wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 11, pp. 1–15, 2016.
[35] Y. Shen, L. S. Huang and W. Yang, "Achieving personalized and privacy-preserving range queries over outsourced cloud data," in *IEEE Int. Conf. on Communications*, Paris, France, pp. 1–6, 2017.
[36] R. Lu, "A new communication-efficient privacy-preserving range query scheme in fog-enhanced IoT," *IEEE Internet on Things Journal*, vol. 6, no. 2, pp. 2497–2505, 2018.
[37] D. Boneh, J. E. Goh and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography Conf.*, Berling, Germany, pp. 325–341, 2005.
[38] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Int. Conf. on the Theory and Applications of Cryptographic Techniques*, Berlin, Germany, pp. 223–238, 1999.
[39] M. S. Islam, M. Kuzu and M. Kantarcioglu, "Interface attack against encrypted range queries on outsourced databases," in *Proc. of the 4th ACM Conf. on Data and Application Security and Privacy*, New York, USA, pp. 235–246, 2014.
[40] M. Naveed, S. Kamara and C. V. Wright, "Interface attacks on property-preserving encrypted databases," in *Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security*, New York, USA, pp. 644–655, 2015.
[41] G. Kellaris, G. Kollios, K. Nissim and A. Neill, "Generic attacks on secure outsourced databases," in *Proc. of the ACM SIGSAC Conf. on Computer and Communications Security*, New York, USA, pp. 1329–1340, 2016.
[42] M. Lacharite, B. Minaud and K. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in *IEEE Symp. on Security and Privacy*, San Francisco, USA, pp. 297–314, 2018.
[43] C. A. De and V. Iovino, "jPBC: Java pairing based cryprography," in *IEEE Symp. on Computers and Communication*, Kerkyra, Greece, pp. 850–855, 2011.
[44] B. Huang and S. Liang, "A range search scheme based on encrypted index hiding order and access patterns," in *Int. Conf. on Internet of Things (iThings)*, Atlanta, USA, pp. 340–347, 2019.
[45] C. Guo, R. Zhuang, Y. Jie, K. Choo and X. Tang, "Secure range search over encrypted uncertain IoT outsourced data," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1520–1529, 2018.
[46] J. Liang, Z. Qin, S. Xiao, J. Zhang, H. Yin *et al.,* "Privacy-preserving range query over multi-source electronic health records in public clouds," *Journal of Parallel and Distributed Computing*, vol. 135, no. 4, pp. 127–139, 2020.