Peer-to-Peer Networking and Applications (2021) 14:2156–2170
https://doi.org/10.1007/s12083-021-01179-2

Efficient and privacy-preserving range-max query in fog-based agricultural IoT

Min Zhou[1,2] · Yandong Zheng[2] · Yunguo Guan[2] · Limin Peng[1] · Rongxing Lu[2]

Received: 12 October 2020 / Accepted: 6 May 2021 / Published online: 25 May 2021

Abstract

Smart agriculture Internet of Things (IoT) is a typical application of IoT and has become popular due to its advantages in automatic irrigation and fertilization, crop growth monitoring, pest and disease detection, etc. To reduce resource waste, minimize environmental impact, and maximize crop yield, most smart agricultural applications require to collect and process agricultural data in real-time. However, the computational and storage resources of the agricultural IoT devices are limited. To alleviate the computational and storage pressure on agriculture IoT devices and timely process the collected data collected by IoT devices, the fog node is usually placed at the edge of the agricultural IoT. Nevertheless, the fog node may not be completely trusted. The agricultural IoT devices' data stored in the fog node will face the potential risk of privacy leakage. In this paper, to preserve the privacy of agricultural IoT devices' data and user query's result in the fog-based smart agriculture IoT, we first build the K^2-treap, which is used for storing the data collected by agriculture IoT devices and support efficient range-max query and dynamic update of the data. Then, we design a data encryption and comparison algorithm based on BGN homomorphic encryption technique and present an efficient and privacy-preserving range-max query in the fog-based smart agriculture IoT, which can not only securely compare two data based on their ciphertexts but also support the incremental update directly over ciphertexts. Notably, our comparison technique and range-max queries are run by the fog node, so there are no interactions between the agricultural IoT devices and the fog node during the comparison and query. Finally, we conduct a detailed security analysis and performance evaluation. The results show that our proposed scheme can indeed protect the privacy of the agricultural IoT devices' data and query results, and the experimental test results prove that our proposed scheme is efficient.

Keywords Privacy-preserving · Range-max query · Fog · Agricultural IoT

1 Introduction

Smart agriculture IoT has become popular due to its advantages in automatic irrigation and fertilization, crop growth monitoring, and pest detection, etc [1, 2]. It is a typical application of IoT and is mostly the same as the traditional IoT in terms of architecture, protocol standards, wireless communication technology. Differently, to reduce waste of resources, minimize environmental impact, and maximize crop yields, most smart agricultural applications require to collect and process agricultural data in real-time. Meanwhile, the accurate geographical location and the geographical dimension of species are very important, which is a critical link in the production and circulation of agricultural products.

With the widespread application of smart agriculture, the amount of agricultural data perceived by IoT devices will greatly increase. However, the computational and storage resources of the IoT devices are limited. To alleviate the computational and storage pressure on IoT devices and timely process the collected data, fog nodes are usually placed at the edge of the agricultural IoT to store data and process queries over the stored data [3–7]. Nevertheless, the fog node is at the edge of the agricultural IoT, and may not be completely trusted. If the data collected by the IoT device is directly exposed to the fog node, it will pose a privacy threat to the data [8–14]. These potential privacy threats may result in an unproductive farming environment. For example, if the smart agriculture system is not working correctly, it could make a wrong decision based

✉ Min Zhou
zmfw@scau.edu.cn

Extended author information available on the last page of the article.

on the wrong data, such as over-fertilizing, or not irrigating dry land.

Some researches on privacy protection have been conducted in the fields of energy, healthcare, or transportation [15–19]. However, few studies have focused on privacy threats in agriculture. To protect the privacy of agriculture data, the commonly used method is that the IoT devices first encrypt their collected data and then report them to the fog node. However, the encryption technique will inevitably affect data query functions. Among these query functions, the range-max query is one of the most common queries. The range-max query is to obtain the maximum data in a specific region.

For clarity, we give an example of the range-max query as follows:

Example 1 Suppose there are 4×4 agriculture IoT devices deployed in the whole field to monitor the growth of crops. Each IoT device is responsible for collecting crop growth data and reporting the data and its location to the fog node. And then, the fog node receives 16 triples, as shown in Fig. 1. For example, (16, 0, 0) represents that the data of the crop growth collected by the IoT device at location (0, 0) is 16. When the query user wants to know the crop that grows best in the rectangular range $Q = [1, 2] \times [1, 2]$ marked with a red border in Fig. 1, he/she sends the query range Q to the fog node and will receive the maximum value 20 of the location (2, 1) in that range returned by the fog node.

However, the range-max query over the ciphertext data is more challenging than that over plaintext data. To the best of our knowledge, existing works focus on either privacy-preserving max/min query or range query [20–26]. No previous work has addressed the privacy-preserving range-max queries in smart agriculture IoT. In response to this challenge, we present an efficient privacy-preserving range-max query scheme in fog-based smart agriculture IoT. The main contributions of this paper are fourfold as follows.

- First, we build the K^2-treap which be used to store the data collected by agriculture IoT devices and support efficient range-max query and dynamic data update.

(16,0,0)	**(15,0,1)**	**(5,0,2)**	**(9,0,3)**
(14,1,0)	**(13,1,1)**	**(4,1,2)**	**(1,1,3)**
(10,2,0)	**(20,2,1)**	**(2,2,2)**	**(21,2,3)**
(19,3,0)	**(11,3,1)**	**(3,3,2)**	**(8,3,3)**

Fig. 1 An example of the range-max query

- Second, we design a data encryption and comparison algorithm based on BGN homomorphic encryption technique, which can not only securely compare two data based on their ciphertexts but also support the incremental updates directly over ciphertexts.
- Third, based on the K^2-treap and our data encryption and comparison algorithm, we present an efficient privacy-preserving range-max query scheme. Notably, our comparison technique and range-max queries are run by the fog node, so there are no interactions between the agricultural IoT devices and the fog node during the comparison and query.
- Finally, we conduct a detailed security analysis, and the results show that the scheme we proposed can indeed protect privacy. Through performance evaluation, the experimental test results prove that our proposed scheme is efficient.

The rest of this paper is arranged as follows. In Section 2, we describe our models, namely the system model and the security model, and the design goals. In Section 3, we introduce the preliminary. Then, we present our proposed scheme in Section 4. In Section 5 and Section 6, we conduct security analysis and performance evaluation respectively. Some related works are discussed in Section 7. Finally, we summarize our work and clarify the content of future research in Section 8.

2 Our models and design goals

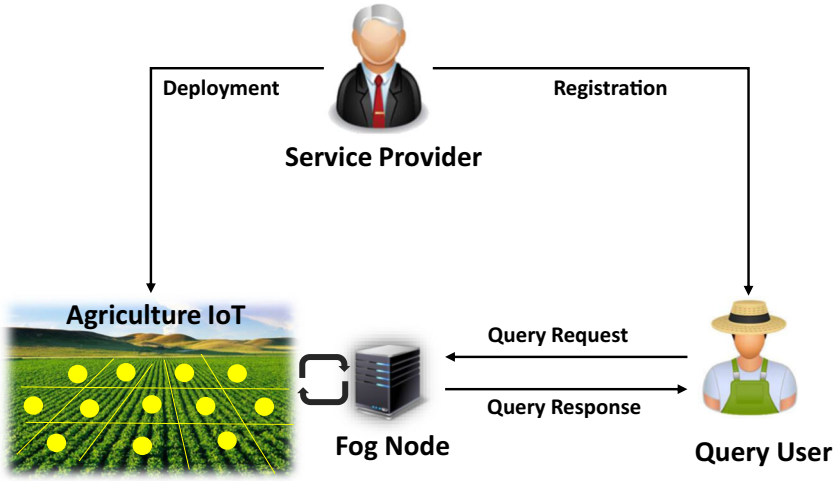
In this section, we first describe our models, namely the system model and the security model, and then propose our design goals.

2.1 System model

Our system model with the privacy-preserving range-max query in the fog-based agricultural IoT is mainly composed of four types of entities, namely, service provider, agricultural IoT devices, fog node, and query user, as shown in Fig. 2.

- **Service provider:** The service provider is responsible for the whole system. To remotely monitor crop growth in the field, The service provider divides the filed evenly into $2^n \times 2^n$ equal cells and deploys the agriculture IoT devices in each cell. Due to the limited storage space and computational capacity of the IoT devices, the service provider deploys a fog node at the edge of the agricultural IoT to help the IoT device complete data storage and processing. The data stored in the fog node is available for query from those users authorized by the service provider.

Fig. 2 System model



- **Agriculture IoT devices:** In the system, there are $2^n \times 2^n$ IoT devices, and they are distributed in the whole field to monitor the growth of the crops. In specific, the IoT device $I_{x,y}$ is distributed in the location (x, y) and responsible for collecting the data $D_{x,y}$ of the cell (x, y), where $0 \leq x, y \leq 2^n - 1$ and $D_{x,y} \in [0, l]$. Let $(D_{x,y}, x, y)$ denote a triple tuple of the IoT device $I_{x,y}$. Then, the data collected by all IoT devices can be denoted as $\mathbb{D} = \{(D_{x,y}, x, y) | 0 \leq x, y \leq 2^n - 1)\}$. To prevent the fog node from knowing the collected data, each $I_{x,y}$ encrypts the data $D_{x,y}$ before sending it to the fog node. Thus, the fog node can receive an encrypted dataset $[\![\mathbb{D}]\!] = \{([\![D_{x,y}]\!], x, y) \, | 0 \leq x, y \leq 2^n - 1\}$ from all IoT devices, where $[\![D_{x,y}]\!]$ denotes the ciphertext of $D_{x,y}$.
- **Fog node:** When fog node receives encrypted dataset $[\![\mathbb{D}]\!] = \{([\![D_{x,y}]\!], x, y) \, | 0 \leq x, y \leq 2^n - 1\}$ from all IoT devices, it can provide the range-max query service to query users. Specifically, when the fog node receives a range-max query request in the form of $Q = [x_1, x_2] \times [y_1, y_2]$, it will search the encrypted dataset and return the encrypted maximum data, i.e., $[\![D_{max}]\!]$, within the query rectangular range $Q = [x_1, x_2] \times [y_1, y_2]$, where $D_{max} = \max\{D_{x,y} | x_1 \leq x \leq x_2; y_1 \leq y \leq y_2\}$. In addition, the fog node allows agriculture IoT devices to dynamically update their data in ciphtexts. For example, when the agriculture IoT device $I_{x,y}$ collects a new incremented data $D'_{x,y}$ at location (x, y), and it can send corresponding encrypted data $[\![D'_{x,y}]\!]$ to the fog node. Then, the fog node will calculate $[\![D_{x,y}]\!] \cdot [\![D'_{x,y}]\!]$ to obtain $[\![D_{x,y} + D'_{x,y}]\!]$ and use $[\![D_{x,y} + D'_{x,y}]\!]$ to update the value $[\![D_{x,y}]\!]$ on the location (x, y) node.
- **Query user:** The user must first register with the service provider to access the fog node. Then the service provider will assign a key to the registered user. Only those users authorized by the service provider can access the fog node for data queries. When the query user wants to know the crop that grows best in the rectangular range $Q = [x_1, x_2] \times [y_1, y_2]$, he/she sends the query range Q to the fog node and receives the maximum value $[\![D_{max}]\!]$ in that range and the corresponding location (x_{max}, y_{max}), returned by the fog node. Finally, The query user uses the key to recover the D_{max} from the encrypted $[\![D_{max}]\!]$.

2.2 Security model

In our security model, the service provider is considered *trusted* and is responsible for system initialization and key distribution. For the agriculture IoT devices, they are considered *honest*, and always keep in good condition, i.e., they will follow the protocol sincerely and send the real encrypted value and the correct location $[\![\mathbb{D}]\!] = \{([\![D_{x,y}]\!], x, y) | 0 \leq x, y \leq 2^n - 1\}$ to the fog node. For the fog node, it is considered *honesty-but-curious*. That is, it will comply with the protocol to store or update the encrypted data $[\![D_{x,y}]\!]$ at the location (x, y), and run query algorithm correctly to respond to the authorized user's range-max query. However, it may be curious about some private information, such as the plaintext of values from agriculture IoT devices and the corresponding query results of the authorized user. For the query users, they are *honest*, i.e., they will honestly comply with the protocol to submit the range-max queries. Besides, we assume that there is no collusion between query users and the fog node. This assumption is reasonable because the penalty of collusion for the involving fog node is very high, including losing the trust of users and being prosecuted.

Of course, there may be active attacks such as damage to IoT devices. Since we are focusing on protecting privacy in this paper, the active attacks are no longer within the scope of this article, and it will be our work in the future.

2.3 Design goals

Based on the system model described in Section 2.1 and the security model described in Section 2.2, we present an efficient and privacy-preserving range-max query scheme. In particular, the following two goals should be achieved.

- **Privacy protection:** Privacy protection is the basic requirement of our paper. That is to say, the data collected by IoT devices and the corresponding query results of query users should be kept confidential from the fog node.
- **Dynamic update:** In addition to privacy protection requirements, the proposed scheme should also support the dynamic update of IoT devices' data in the ciphertext. Specifically, when the data collected by IoT devices $I_{x,y}$ changes, the proposed scheme should support the fog node to dynamically update corresponding encrypted data $[\![D_{x,y}]\!]$ stored in it.

3 Preliminary

In this section, we will outline some knowledge related to our proposed scheme, namely the bilinear pairing with composite order and the BGN homomorphic encryption[27].

3.1 Bilinear pairing with composite order

Let $N = pq$, where p, q are two different large prime numbers of the same length. $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups with composite order N. The group can be instantiated by the elliptic curve addition group. The bilinear map can be computed in polynomial time. We assume that the discrete logarithm problems in both $\mathbb{G}$ and $\mathbb{G}_T$ are hard.

If the map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties, this map is called *bilinear map with composite order*.

- **Bilinearity**: For any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}_N$, we have $e(g^a, h^b) = e(g, h)^{ab}$;
- **Non-degeneracy**: There is a generator $g \in \mathbb{G}$, so that $e(g, g)$ is with the order N in $\mathbb{G}_T$. At the same time, $e(g, g)$ is a generator of $\mathbb{G}_T$.
- **Computability**: For all $(g, h) \in \mathbb{G}$, there is an algorithm that can efficiently compute $e(g, h) \in \mathbb{G}_T$.

Definition 1 Composite Bilinear Generator

Let $\mathcal{CG}en$ be a composite bilinear parameter generator, and κ is a security parameter. According to the following process, the $\mathcal{CG}en$ can output a 5-tuple $(N, \mathbb{G}, \mathbb{G}_T, g, e)$ in probability polynomial time.

(1) Let $N = pq$, where p, q are two randomly generated κ-bit prime numbers.
(2) Generate two groups $\mathbb{G}, \mathbb{G}_T$ with order N, and select a generator g for $\mathbb{G}$. Then, construct a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ as mentioned above.
(3) Output the parameters $(N, \mathbb{G}, \mathbb{G}_T, g, e)$.

Let g be a generator of $\mathbb{G}$, then $\mathbf{g} = g^q \in \mathbb{G}$ can construct the subgroup $\mathbb{G}_p = \{\mathbf{g}^0, \mathbf{g}^1, \cdots, \mathbf{g}^{p-1}\}$ with order p, and $\mathbf{g}' = g^p \in \mathbb{G}$ can construct the subgroup $\mathbb{G}_q = \{\mathbf{g}'^0, \mathbf{g}'^1, \cdots, \mathbf{g}'^{q-1}\}$ with order q in $\mathbb{G}$.

Given a tuple $(N, \mathbb{G}, \mathbb{G}_T, e, h)$, determining whether h is in the subgroup $\mathbb{G}_q$ is called the *SubGroup Decision (SGD) Problem*, where $h \in \mathbb{G}$ or $h \in \mathbb{G}_q$. Since the SGD problem is hard [28], it can be ensured that the following BGN homomorphic encryption is safe.

3.2 BGN Homomorphic encryption

The BGN is a popular homomorphic public-key encryption that was originally proposed by Boneh, Goh, and Nissim in [27]. It includes three phases: key generation, encryption, and decryption.

- **Key Generation:** Let κ be the security parameter, the generator $\mathcal{CG}en$ outputs a 5-tuple $(N, \mathbb{G}, \mathbb{G}_T, g, e)$, where the 5 parameters are described as above. Let $h = g^q$, and it can construct a subgroup of $\mathbb{G}$ of order p. That is, h is a random generator. Therefore, the public key is $pk = (N, \mathbb{G}, \mathbb{G}_T, g, e, h)$, and the private key is $sk = p$.
- **Encryption:** Assuming that the message space is $\mathbb{S} = \{0, 1, \cdots, T\}$, where $T \ll q$. Use $C = E(m) = g^m h^r \in \mathbb{G}$ to encrypt a message $m \in \mathbb{S}$, where $r \in \mathbb{Z}_N$ is a random number.
- **Decryption:** Given a ciphertext $C = E(m)$, the plaintext m can be recovered in the following two steps.
  (1) Compute $C^p = (g^m h^r)^p = (g^p)^m$.
  (2) Let $\hat{g} = g^p$, recover m from $(g^p)^m$ by computing the discrete log of C^p base $\hat{g}$ with the Pollard's lambda method [29].

Based on the $SubGroup\ Decision(SGD)\ Problem$ assumption, BGN is provably secure against chosen-plaintext attacks. For the detailed security analysis please refer to [27]. BGN encryption has the following homomorphic properties.

- **Addition**: Given $E(m_1)$ and $E(m_2)$, we have $E(m_1) \cdot E(m_2) \to E(m_1 + m_2)$.
- **Multiplication-I**: Given $E(m_1)$ and m_2, we have $E(m_1)^{m_2} \to E(m_1 \cdot m_2)$.

– **Self-Blinding**: Given $E(m_1)$ and h^{r_2}, we have $E(m_1) \cdot h^{r_2} \rightarrow E(m_1)$.
– **Multiplication-II**: Given $E(m_1)$ and $E(m_2)$, we have $e(E(m_1), E(m_2)) \rightarrow E'(m_1 \cdot m_2) \in \mathbb{G}_T$, where $E'()$ represents a ciphertext in $\mathbb{G}_T$.

4 Our proposed scheme

In this section, we present our efficient and privacy-preserving range-max query scheme in fog-based agricultural IoT. Before going into details, we first introduce a tree structure, called K^2-treap, and a privacy-preserving encryption scheme, a privacy-preserving comparison protocol, which are several important components of our proposed scheme.

4.1 The K^2-treap

The K^2-treap is a quadtree structure and also has the max heap property. Specifically, each internal node in the K^2-treap has four children, and the value of child nodes is less than that of their parent node. With the max heap property, the K^2-treap can effectively support the range-max query. In the following, we will introduce three algorithms about the K^2-treap including tree building, dynamic update, and range-max query.

4.1.1 Tree building

Suppose that the dataset $\mathbb{D}$ is a location-based dataset, and it is in the form of $\mathbb{D} = \{(D_{x,y}, x, y) | 0 \leq x, y \leq 2^n - 1)\}$, where (x, y) is a location and $D_{x,y}$ is the value in (x, y), as shown in Fig. 3. Based on the dataset $\mathbb{D}$, we can build a K^2-treap K_T from bottom to top as follows.

- ***Step 1:*** Build the leaf nodes of the K^2-treap K_T. Specifically, for each value $D_{x,y} \in \mathbb{D}$, build a leaf node $(D_{x,y}, x, y)$. Thus, the side length of the area covered by each leaf node can be denoted as $L = 2^0$.
- ***Step 2:*** Build the higher level of the K^2-treap K_T based on the leaf nodes. As shown in Fig. 3, from the upper-left node of the whole area, we can gradually merge four leaf nodes in a square into an internal node. In the internal node, it contains 7 attributes $\{\text{AID}, \text{CID}, c_1, c_2, c_3, c_4, p\}$. AID is used to describe the area that is covered by the current node. The area is a square and it can be denoted as $\text{AID} = (x_{left}, y_{top}, L)$, where (x_{left}, y_{top}) is the upper-left location of the area, and L is the side length of the area. In this level, since there are 2×2 values in the area of the square, the side length of the area is $L = 2^1$. CID is used to describe the max value and its corresponding location of this area. It can be denoted as $\text{CID} = (data, x, y)$, where $data$ is the max value in this square, i.e., $data = \max\{D_{x,y} | x \in [x_{left}, x_{left} + L], y \in [y_{top}, y_{top} + L]\}$, and (x, y) is the location of $data$. For c_1, c_2, c_3, c_4, they are pointers to the children of the current internal node. Without loss of generality, we assume that c_1, c_2, c_3, c_4 respectively point to the upper-left node, upper-right node, lower-left node, and lower-right node of this area. Meanwhile, the internal node has a pointer p that points to its parent, and it will be assigned when the next higher level is built. When constructing this level, we let the parent pointers of the leaf nodes in this area point to the current internal node.
- ***Step 3:*** Build the next higher level of the K^2-treap K_T based on the internal nodes constructed in step 2. From the upper-leaf node, gradually merge four internal nodes in a square into an internal node and assign the attributes of the internal node with the same method

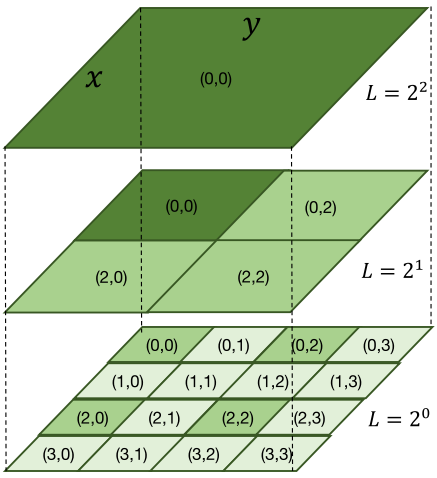


Fig. 3 The K^2-treap structure

Area ID AID=(aleft-x, atop-y, L)		Parent= null	
Cell ID with Max Value in AID, CID=(cleft-x, ctop-y)		Data=null	
Child00 =null	Child01 =null	Child10 =null	Child11 =null

in step 2. Differently, in this level, the area covered by each internal node contains $2^2 \times 2^2$ values, the side length of the square is $L = 2^2$.

- ***Step 4:*** Repeat step 3 until there is only one merged node, i.e., the root node. After this step, a K^2-treap K_T is constructed.

Example 2 There is a dataset with 4×4 elements. We can use the dataset to build the K^2-treap K_T, as shown in Fig. 4. The building process of the K_T is as follows:

- ***Step 1:*** Build the leaf nodes of the K_T. Each element of the dataset can be used to build the leaf node of the K_T. At the same time, the side length of the area is $L = 2^0$.
- ***Step 2:*** Build the higher level of the K_T based on the leaf nodes. From the upper-left node of the whole area, we gradually merge four leaf nodes in a square into an internal node, and obtain 2×2 internal nodes. The value of each internal node is the maximum data of the current square. The CID of the four internal nodes are $(16, 0, 0)$, $(9, 0, 3)$, $(20, 2, 1)$, $(21, 2, 3)$.The AID of the four internal nodes are $(0, 0, L = 2^1)$, $(0, 2, L = 2^1)$, $(2, 0, L = 2^1)$, $(2, 2, L = 2^1)$. Each internal node's four child-pointers c_1, c_2, c_3, c_4 respectively point to the upper-left node, upper-right node, lower-left node, and lower-right node of this area. At the same time, the side length of the area is $L = 2^1$.
- ***Step 3:*** Build the next higher level of the K_T based on the internal nodes constructed in Step 2, we merge the four internal nodes in a square. There is only one merged node, which is the root node of the K_T. The AID of the root node is $(0, 0, L = 2^2)$, the CID of the root node is $(21, 2, 3)$. In this case, we can get the maximum data 21 of the whole dataset and the corresponding location $(2, 3)$ from the CID of the root node.

Algorithm 1 Range-max query over the K^2-treap.

Input: The K^2-treap K_T;
The query range $Q = [x_1, x_2] \times [y_1, y_2]$;
Output: The query result D_{max} and its location (x_{max}, y_{max});

```
1: Stack S = ∅;
2: int D_max=0, (x_max, y_max)=(0,0);
3: S.push(K_T.root);
4: while S is not empty do
5:     N = S.pop();
6:     Let N's CID is (data, x, y);
7:     if (x, y) ∈ Q then
8:         if D_max < data then
9:             D_max = data
10:            (x_max, y_max) = (x, y);
11:    else
12:        Let c_1, c_2, c_3, c_4 be N's children;
13:        for each c_i do
14:            if N.c_i.AID has intersection with Q then
15:                S.push(N.c_i);
16: return D_max, (x_max, y_max);
```

4.1.2 Range-max Query

When the K^2-treap K_T is built, it can efficiently support range-max query. In specific, given a query range $Q = [x_1, x_2] \times [y_1, y_2]$, we can search K_T to find out the maximum value among all values within the range Q, i.e., $D_{max} = \max\{D_{x,y} | (x, y) \in Q\}$. The query process can be performed as the following steps.

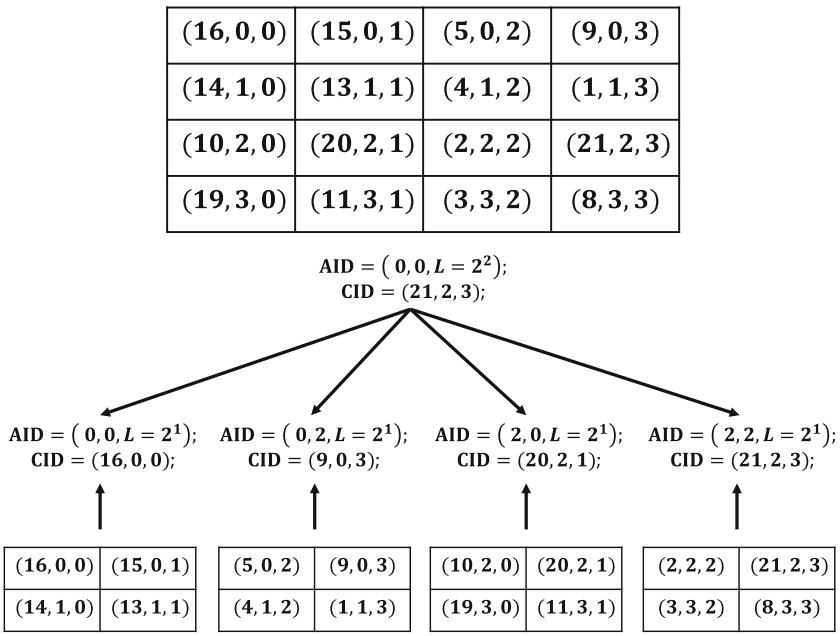


Fig. 4 An example of K^2-treap building

- ***Step 1:*** Initialize a stack S $= \emptyset$, $D_{max} = 0$, $(x_{max}, y_{max}) = (0, 0)$. The stack will be used to store the selected nodes whose AID intersects with Q. D_{max} represents the maximum value in the range Q, (x_{max}, y_{max}) is the location where the maximum value locates in.
- ***Step 2:*** Push the root node of the K_T into the stack.
- ***Step 3:*** If the stack is not empty, pop a node from the stack, and denote it by N. If the location (x, y) of the N.CID is in the query range Q and the data N.CID.$data$ is larger than D_{max}, then update D_{max} and (x_{max}, y_{max}). If the location (x, y) of the node N.CID is not in the query range Q, we select its child-nodes whose AID has an intersection with Q and push them into the stack S.
- ***Step 4:*** Repeat *Step 3*, until the stack is empty.

Note that, the query area Q in the algorithm is a regular rectangle, the algorithm can query the maximum value of any irregular region, as any irregular region can ultimately be made up of multiple cells.

4.1.3 Dynamic update

When the value $D_{x,y}$ of the node N at the location (x, y) in the K^2-treap K_T has changed, the incremented value is $D'_{x,y}$. The K_T can be dynamically updated by comparing the value of the node with the value of its parent. The dynamical update process of the K_T is as follows:

- ***Step 1:*** Jump from the root node to the corresponding location of the node N in the leaf node according to the (x, y);
- ***Step 2:*** Update the data $D_{x,y}$ of the node's N.CID with new data $D_{x,y} + D'_{x,y}$.
- ***Step 3:*** Compare the data N.CID.$data$ with the data $N.p$.CID.$data$, If the data N.CID.$data$ $>$ $N.p$.CID.$data$, then update the $N.p$.CID.$data$ with the N.CID.$data$, and use N to represent the parent node.
- ***Step 4:*** Repeat *Step 3*, until the node is root or N.CID.$data \leq N.p$.CID.$data$.

4.2 The data encryption and comparison technique

In this subsection, we present data encryption and comparison technique, which can achieve data encryption and encrypted data comparison. This technique is based on BGN homomorphic encryption technique, and it involves three parties, i.e., data encryptor, data decryptor, and data comparator. The data encryptor is responsible for encrypting the data. The data comparator can achieve encrypted data comparison. The decryptor can recover plaintexts from encrypted data. In specific, our data encryption and data comparison protocol technique consist of four algorithms, i.e., key generation, encryption, decryption, and data comparison.

Algorithm 2 Dynamic update the K^2-treap.

Input: The K^2-treap ;
The increasing value $D'_{x,y}$ at location (x, y).
Output: The updated K^2-treap.

```
Let N represent the root of K^2-treap
N.CID is (data, x, y);
N.AID is (x, y, L);
while N.AID.(x, y) ≠ (x, y) and N.AID.(x, y) ≠ N.CID.(x, y) do
    Let c1, c2, c3, c4 be N's children;
    for i = 1 to 4 do
        Pick N.ci which AID has an intersection with (x, y)
        N=N.ci;
N.CID.data = D_{x,y} + D'_{x,y}.
while N is not the root node, and N.CID.data > N.p.CID.data do
    N.p.CID = N.CID ;
    N =N.p;
```

– **Key generation:** Let κ be the security parameter, the key generation algorithm first outputs a 5-tuple $(N, \mathbb{G}, \mathbb{G}_T, g, e)$ by running $\mathcal{GG}en(\kappa)$, where $N = pq$. Then, let $h = g^q$, the BGN public key is $pk = (N, \mathbb{G}, \mathbb{G}_T, g, e, h)$, and the private key is $sk = p$. Further, the algorithm selects a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$, and also selects two secret keys $s, t \in \mathbb{Z}_N$. In addition, to enable the data comparator to compare encrypted data, the key generation algorithm generates a set of hash values $\mathbb{H} = \{H(e(g, g)^{pstx})|x \in (0, l]\}$. Finally, the key generation algorithm respectively distributes g^s, $(s^{-1}p)$, and $(g^{pt}, \mathbb{H})$ to the encryptor, decryptor, and data comparator. Meanwhile, it publishes the public key pk, and hash function H.
– **Encryption:** Given a plaintext $m \in (0, l]$, the data encryptor can encrypt m as

$$[\![m]\!] = E(m) = (g^s)^m \cdot h^r \tag{1}$$

where $r \in \mathbb{Z}_N$ is a random number.
– **Decryption:** Given a ciphertext $[\![m]\!] = E(m)$, the decryptor first computes

$$[\![m]\!]^{s^{-1}p} = ((g^s)^m \cdot h^r)^{s^{-1}p} = (g^p)^m. \tag{2}$$

Then, the decryptor can recover m from $(g^p)^m$ by computing the discrete log of $(g^p)^m$ base g^p with the Pollard's lambda method [29].

- **Data comparison:** Given two encrypted data $[\![m_1]\!]$ and $[\![m_2]\!]$, the data comparator can compare them. Specifically, the data comparator first computes

$$X = e\left(\frac{[\![m_1]\!]}{[\![m_2]\!]}, g^{pt}\right) = e(g, g)^{pst(m_1 - m_2)} \tag{3}$$

Then, if $X = 1_{\mathbb{G}_T}$, we can directly have $m_1 = m_2$. If $H(X) \in \mathbb{H}$, $m_1 > m_2$. Otherwise, $m_1 < m_2$.

Correctness. The correctness of the data encryption and comparison technique is follows:

Assume $[\![m_1]\!]$ and $[\![m_2]\!]$ are two encrypted values,

$$\frac{[\![m_1]\!]}{[\![m_2]\!]} = \frac{(g^s)^{m_1} \cdot h^{r_1}}{(g^s)^{m_2} \cdot h^{r_2}} = g^{s(m_1-m_2)} \cdot h^{r_1-r_2}$$

Then, we have

$$X = e\left(\frac{[\![m_1]\!]}{[\![m_2]\!]}, g^{pt}\right) = e(g^{s(m_1-m2)} \cdot h^{r_1-r_2}, g^{pt})$$
$$= e(g, g)^{pst(m_1-m_2)} \cdot e(h, g)^{pt(r_1-r_2)}$$
$$= e(g, g)^{pst(m_1-m_2)} \cdot 1 = e(g, g)^{pst(m_1-m_2)}$$

Clearly, if $m_1 - m_2 > 0$, i.e. $m_1 > m_2$, then $m_1 - m_2 \in (0, l]$ and we know $H(X)$ will appear in $\mathbb{H}$. if $X = 1_{\mathbb{G}_T}$, we can directly have $m_1 = m_2$. If $m_1 < m_2$, $H(X_i)$ will not appear in $\mathbb{H}$. Therefore, the correctness of the data encryption and comparison technique holds.

4.3 The details of our scheme

In this subsection, we present our privacy-preserving range-max query scheme, which contains four phases, i.e., system initialization, data transmission, range-max query, and dynamic update.

4.3.1 System initialization

In the system initialization, the service provider is responsible for bootstrapping the whole scheme.As the key generation algorithm in Section 4.2, the service provider generates the public pk and the private key sk, where $pk = (N, \mathbb{G}, \mathbb{G}_T, e, g, h)$, $sk = p$. Further, the service provider chooses a cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$, and two secret keys $s, t \in \mathbb{Z}_N$. Finally, the service provider keeps (p, s, t) secretly, respectively distributes g^s, $(s^{-1}p)$, and $(g^{pt}, \mathbb{H})$ to the IoT devices, the authorized query use, and fog node. Meanwhile, it publishes the public key pk, and hash function H. The hash function is constructed as described in Section 4.2.

4.3.2 Data transmission

In this phase, The IoT device transmits its data to the fog node as follows:

- ***Step 1:*** IoT device $I_{x,y}$ uses g^s and a random number $r_{x,y} \in \mathbb{Z}_N$ to encrypt $D_{x,y}$ as

$$[\![D_{x,y}]\!] = E(D_{x,y}) = (g^s)^{D_{x,y}} \cdot h^{r_{x,y}} \quad (4)$$

where $r_{x,y} \in \mathbb{Z}_N$, $0 \leq x, y \leq 2^n - 1$, $D_{x,y} \in \mathbb{D}$.

- ***Step 2:*** IoT device $I_{x,y}$ sends the encrypted data $[\![D_{x,y}]\!]$ and corresponding location (x, y) to the fog node.
- ***Step 3:*** From all IoT devices, the fog node receives an encrypted dataset $[\![\mathbb{D}]\!] = \{([\![D_{x,y}]\!], x, y) | 0 \leq x, y \leq 2^n - 1)\}$. Based on the encrypted dataset, the fog node builds a K^2-treap K_T as described in Section 4.1.1. Since the fog node cannot access the plaintext dataset $\mathbb{D}$, it will face a challenge, that is how to compare the sizes of these encrypted data. For solving the challenge, the fog node can run privacy-preserving data comparisons described in Section 4.2. In this case, the node in the K_T contains 7 attributes {AID, CID, c_1, c_2, c_3, c_4, p} as described in Section 4.1.1. The only difference is that the data of the CID is encrypted.

4.3.3 Range-max query

When the K_T is built by fog node, it can efficiently support range-max query over the encrypted data as follows:

- ***Step 1:*** If an authorized query user wants to know which crops grow best in a range, it needs to send the query range Q to the fog node, where $Q = [x_1, x_2] \times [y_1, y_2]$.
- ***Step 2:*** The fog node runs the range-max query algorithm as described in Section 4.1. Due to the data of the node's CID in the K_T is encrypted, it is a challenge for the fog node to compare $[\![D_{max}]\!] < [\![N.\text{CID}.data]\!]$. In this case, the fog node needs to run privacy-preserving data comparisons described in Section 4.2, then returns the maximum encrypted value$[\![D_{max}]\!]$and corresponding location x_{max}, y_{max} to the authorized query user. i.e., $[\![D_{max}]\!] = \max\{[\![D_{x,y}]\!] | (x, y) \in Q\}$.
- ***Step 3:*** Once the query user receives the value$[\![D_{max}]\!]$ and location x_{max}, y_{max}, he/she can use the key $(s^{-1}p)$ assigned by the service provider to perform the following calculation.

$$([\![\mathbb{D}_{max}]\!])^{s^{-1}p} = ((g^s)^{D_{max}} \cdot h^r)^{(s^{-1}p)} = (g^p)^{D_{max}}$$

Let $\hat{g} = g^p$. It can recover D_{max} from $[\![D_{max}]\!]$ by computing the discrete log of $([\![D_{max}]\!])^{s^{-1}p}$ base $\hat{g}$ with the Pollard's lambda method [29].

4.3.4 Dynamic update

When the agriculture IoT device monitors that the value at the location (x, y) has changed, the incremented value is $D'_{x,y}$. The K^2-treap K_T can be dynamically updated by fog node as follows:

- ***Step 1:*** IoT device $I_{x,y}$ first encrypts the value $D'_{x,y}$ using the data encryption algorithm as described in

Section 4.2, then sends the ciphertext $[\![D'_{x,y}]\!]$ and the corresponding location (x, y) to fog node.

- ***Step 2:*** When the fog node receives the encrypted data $[\![D'_{x,y}]\!]$ and location x, y, it first computes $[\![D_{x,y}]\!] \cdot [\![D'_{x,y}]\!]$, then obtains the encrypted data $[\![D_{x,y} + D'_{x,y}]\!]$ according to the additional property of the BGN homomorphic encryption.
- ***Step 3:*** Since the fog node only knows the encrypted data of the node CID, it will face a challenge, that is how to compare $[\![N.\text{CID}.data]\!] > [\![N.p.\text{CID}.data]\!]$, where N represents the node with the value $[\![D_{x,y} + D'_{x,y}]\!]$ at the location (x, y). For solving the challenge, the fog node can run privacy-preserving data comparisons described in Section 4.2 .
- ***Step 4:*** Repeat *Step 3*, until the node is root or $[\![N.\text{CID}.data]\!] \leq [\![N.p.\text{CID}.data]\!]$.

Algorithm 3 Privacy-preserving dynamic update the K^2-treap.

Input: The K^2-treap;
The encrypted value $[\![D'_{x,y}]\!]$ at location (x, y).
Output: The updated K^2-treap.

```
Let N represent the root of K^2-treap
N.CID is (data, x, y);
N.AID is (x, y, L);
while N.AID.(x, y) ≠ (x, y) and N.AID.(x, y) ≠ N.CID.(x, y) do
    Let c1, c2, c3, c4 be N's children;
    for i = 1 to 4 do
        Pick N.ci which AID has an intersection with (x, y)
        N=N.ci;
[[D_{x,y} + D'_{x,y}]] = [[D_{x,y}]] · [[D'_{x,y}]]
N.CID.data = [[D_{x,y} + D'_{x,y}]];
Pick out the encrypted data [[N.CID.data]] and [[N.p.CID.data]];
Perform the "Data comparison" algorithm to compare the two encrypted data;
while N is not the root node, and [[N.CID.data]] > [[N.p.CID.data]] do
    N.p.CID = N.CID ;
    N= N.p;
    Pick out the encrypted data [[N.CID.data]] and [[N.p.CID.data]]
    Perform the "Data comparison" algorithm to compare the two encrypted data;
```

5 Security analysis

In this section, we will conduct the security analysis of our proposed privacy-preserving range-max query scheme. Since the privacy-preserving data encryption and privacy-preserving comparison technique are two important components of our scheme, we first analyze their security. After that, we conduct the security analysis of our proposed scheme.

5.1 Security of data encryption and comparison technique

Our data encryption and comparison technique consist of data encryption and data comparison algorithm. In the following, we show that both of them are secure.

For our data encryption technique, the data is encrypted as $[\![m]\!] = E(m) = (g^s)^m \cdot h^r$. It is very similar to the BGN homomorphic encryption scheme, in which a data m is encrypted as $[\![m]\!] = E(m) = (g)^m \cdot h^r$. Since BGN encryption scheme is semantically secure, our encryption technique is also semantically secure. Meanwhile, compared with BGN encryption technique, our encryption technique involves an additional secret key s, so our scheme is even more secure. Thus, our data encryption technique is secure and it is hard to recover the plaintext m without the private key $s^{-1}p$.

For our data comparison algorithm, given two encrypted data $[\![m_1]\!] = (g^s)^{m_1} \cdot h^r$, $[\![m_2]\!] = (g^s)^{m_2} \cdot h^r$, the data comparator can use the key g^{pt} to compute $X = e\left(\frac{[\![m_1]\!]}{[\![m_2]\!]}, g^{pt}\right) = e(g, g)^{pst(m_1 - m_2)}$. Then, the data comparator can deduce the comparison relationship between m_1 and m_2 by computing $H(X)$ and judging whether $H(X)$ is in the $\mathbb{H}$. If $X = 1_{\mathbb{G}_T}$, the data comparator can directly have $m_1 = m_2$. Actually, the data comparison algorithm pursues the comparison of data on basis of protecting privacy, so the comparison relationship cannot be regarded as private information. However, the data comparator cannot have access to the plaintext of $m_1 - m_2$. This is because the data comparator has no idea on the secret key s, it is hard for it to recover the plaintext $m_1 - m_2$ from $e(g, g)^{pst(m_1 - m_2)}$. Thus, the data comparison algorithm is secure.

5.2 Security of our proposed scheme

In our security model described in Section 2.2, we assume that there is no collusion between the fog node and the query user. That is the query user will not share the key $s^{-1}p$ assigned by the service provider with the fog node. Based on this assumption, we will confirm that our proposed scheme can achieve the privacy-preserving properties, i.e. (1) The dataset $\mathbb{D}$ is privacy-preserving; (2) The query result D_{max} is privacy-preserving; (3) The updated data is privacy-preserving.

– *The dataset $\mathbb{D}$ is privacy-preserving:* In our scheme, IoT device $I_{x,y}$ first encrypts data as $[\![D_{x,y}]\!] = (g^s)^{D_{x,y}} \cdot h^{r_{x,y}}$, $0 \leq x, y \leq 2^n - 1$, and then sends $[\![D_{x,y}]\!]$ to the fog node. Thus, the fog node can access an encrypted dataset $[\![\mathbb{D}]\!]$. Since our data encryption technique is

semantically secure, the fog node can not recover $D_{x,y}$ from $[\![D_{x,y}]\!]$ of the encrypted dataset $[\![\mathbb{D}]\!]$ without knowing the decryption key $s^{-1}p$. As a result, the dataset $\mathbb{D}$ is privacy-preserving.

- *The query result D_{max} is privacy-preserving:* In our scheme, given a query range $Q = [x_1, x_2] \times [y_1, y_2]$, the fog node can find the maximum encrypted value $[\![D_{max}]\!]$ in this query range and return it. Because $D_{max} \in \mathbb{D}$ is encrypted like $[\![D_{max}]\!] = (g^s)^{D_{max}} \cdot h^{r_{x,y}}$, and our data encryption and comparison algorithm is secure, the fog node can not recover the D_{max} from $[\![D_{max}]\!]$ without the decryption key $s^{-1}p$. The D_{max} in range Q is privacy-preserving.
- *The updated data is privacy-preserving:* In our scheme, once the fog node receives a new encrypted incremented value $[\![D'_{x,y}]\!]$ at location (x, y) from the $I(x, y)$, the fog node first computes $[\![D_{x,y}]\!] \cdot [\![D'_{x,y}]\!]$ to obtain the encrypted data $[\![D_{x,y} + D'_{x,y}]\!]$, then uses $[\![D'_{x,y} + D_{x,y}]\!]$ to update the original value $[\![D_{x,y}]\!]$ at the location (x, y) in the K^2-treap. Because $D'_{x,y} \in \mathbb{D}$ is encrypted like $[\![D'_{x,y}]\!] = (g^s)^{D'_{x,y}} \cdot h^{r_{x,y}}$, and our data encryption and comparison algorithm is secure, the fog node can not recover the $D'_{x,y}$ from $[\![D'_{x,y}[]$ without the decryption key $s^{-1}p$. Therefore, the updated data is secret to the fog node.

6 Performance evaluation

In this section, we evaluate the performance of the range-max query scheme by using the time for the fog node to complete the update of K^2 -treap and the time of the range-max query.

To the best of our knowledge, existing works either focus on privacy-preserving range query, top-k query, or max query. No previous work has addressed the privacy-preserving range-max queries in smart agriculture IoT. Therefore we compare our proposed scheme(w/ K^2-treap) with the scheme without integrating the K^2-treap(w/o K^2-treap).

6.1 Experimental setting

We use JAVA to implement our scheme, and run our experiments on a machine with Intel(R) Xeon(R) CPU @2.30GHz, 10GB RAM, and Debian 9 operating system. For our scheme, the corresponding parameters are as follows:

- The encryption algorithm's security parameter κ is 512 bits.
- The length of the p, q is 512-bit, and the length of N is 1024.
- The numbers of the agriculture IoT devices are $2^n \times 2^n$, where $n = 3, 4, 5, 6, 7, 8$.
- The fog node receives an encrypted dataset $[\![\mathbb{D}]\!] = \{([\![D_{x,y}]\!], x, y) | 0 \leq x, y \leq 2^n - 1\}$ from all IoT devices, where $D_{x,y} \in [0, l], l = 500$.

We run our experiments 2^{2n-1} K^2-treap updates and 50 range-max queries.

6.2 Experimental results

6.2.1 Update time

As the scheme without integrating K^2-treap does not involve an update procedure, we mainly focus on the update time for our proposed scheme.

In our experiments, the update of K^2-treap is performed by randomly generating the location (x, y) and the incremented value $0 \leq D'_{x,y} \leq 10$, where $0 \leq x, y \leq 2^n - 1$, $n = 3, 4, 5, 6, 7, 8$. After running 2^{2n-1} times of updating the K^2-treap, the statistical results of the updating time is shown in Fig. 5. In specific, the dots linked by the green lines represent the average update times for varied n, and each violin plot in the diagram has the following properties:

- The bottom and top of the thick black line represent the first and third quartile of the data, respectively. The white circle inside each violin denotes the second quartile of data.
- The lowest (resp. highest) datum within the 1.5 interquartile range (IQR) of the first (resp. third) quartile is marked by the lower (resp. upper) whisker.
- The grey shades show the probability density of the data at different values.

For example, when n=3, there are $2^n \times 2^n = 64$ data collected by the IoT devices. The K^2-treap has 64 leaf nodes and its depth is 4, and we run $2^{2n-1} = 32$ updates on the K^2-treap. Experimental results show that among 32 updates, a quarter of the updates (8 times) can be completed in 551.5ms, half of them (16 times) can be completed in 706.4ms, and three-quarters (24 times) in 987.9ms finish. In Fig. 5, the bottom, white circle, and top of the black box are used to represent them. Meanwhile, the green point represents the average time consumption of 32 updates, which is 759.7 ms.

The other points linked by the green lines represent the average update time when n is 4,5,6,7, and 8. They are $722.9ms$, $763.8ms$, $763.4ms$, $538.8ms$, and $435.9ms$, respectively. The average time consumption for updating the K^2-treap is less than 800 ms.

As shown in Fig. 5, although in rare cases the update time is relatively long. This is mainly because that, the upper bound of update time is linear to the depth of the K^2-treap, which will increase with n. However, as n increases, the possibility of updating the same cell decreases, and

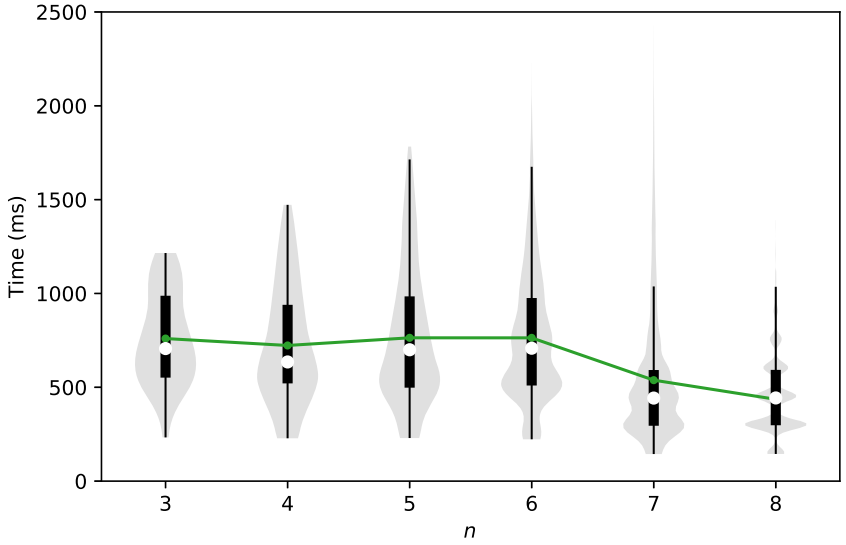


Fig. 5 Update time of K^2-treap. In this figure, each gray shade indicates the update times' distribution for the corresponding value of n, which shows that the upper bound of the update time increases with n. However, the green line shows the average update time, which decreases as n increases

the frequency of updates involving operating internal nodes decreases. Therefore, the average time consumption of the update phase decreases as n increases.

6.2.2 Query time

To compare the query performance, we test the average time consumption for the two schemes to handle queries of varied areas ranging from 0.1×2^{2n} to 1.2×2^{2n}, and for each area of the query range, we randomly generate 50 queries. The average time consumption for maximum queries in different query ranges of the two schemes are shown in Fig. 6.

In the scheme without integrating K^2-treap, as the query range expands, the number of IoT devices within the query range also increases. To find the maximum value in the query range, the compared data will also increase. Therefore, the query time of the scheme will also increase. The query time of the scheme is related to the numbers of data collected by the IoT devices in the query range, therefore the average time consumption of the scheme is linear to the area of the query range.

However, in our scheme(w/ K^2-treap), each internal node in the K^2-treap has four children, and the value of child nodes is less than that of their parent node. With the max heap property, the query time of our scheme will not increase with the increase of the query range. Specifically, when the query range covers the whole field, the data of the root of the K^2-treap is the maximum value in the query range. The query time of our scheme is a very small constant. Therefore our scheme is much better than the scheme without integrating the K^2-treap(w/o K^2-treap). The average time consumption for our proposed scheme reduces as the query range's area increases and is less than 10 seconds.

7 Related works

To the best of our knowledge, existing works focus on either privacy-preserving max/min query [20–22] or range query [23–26], and have little work on privacy-preserving range-max queries in smart agriculture IoT. In the following, we introduce some related works that are close to our work.

Some privacy-preserving max/min query schemes [20–22] have been proposed. Yao et al. [20] are the first to consider the privacy max/min query in two-tiered sensor networks. They use the prefix membership verification approach to encode sensor data. The proposed scheme can not only ensure the storage node to correctly process max/min queries over encoded data but also prevent the storage node from knowing the actual values. Since all sensors in this scheme share the same key, the data privacy could be leaked if a sensor is compromised. Thus, the security of this scheme is weak. Samanthula et al. [21] adopted the GM (Shafi Goldwasser, Silvio Micali) probabilistic encryption to protect data privacy even if a few sensors are compromised. Since every bit of the data collected by a sensor is encrypted into a vector whose length is equal to the domain size of the collected data, the network communication cost may become very large if the domain size of the collected data is large. Thus, this scheme is communication inefficient. Dai et al. [22] are the first to discuss privacy-preserving max/min query (PMQ) processing for the WSN-as-a-Service environment based on the secure multi-party computation. The communication cost of their PMQ scheme is lower than that of the schemes in [20, 21]. Nevertheless, all schemes above have multiple rounds of interactions between sensors during query processing. In addition, these schemes focus on obtaining the max/min value of all encrypted data in the

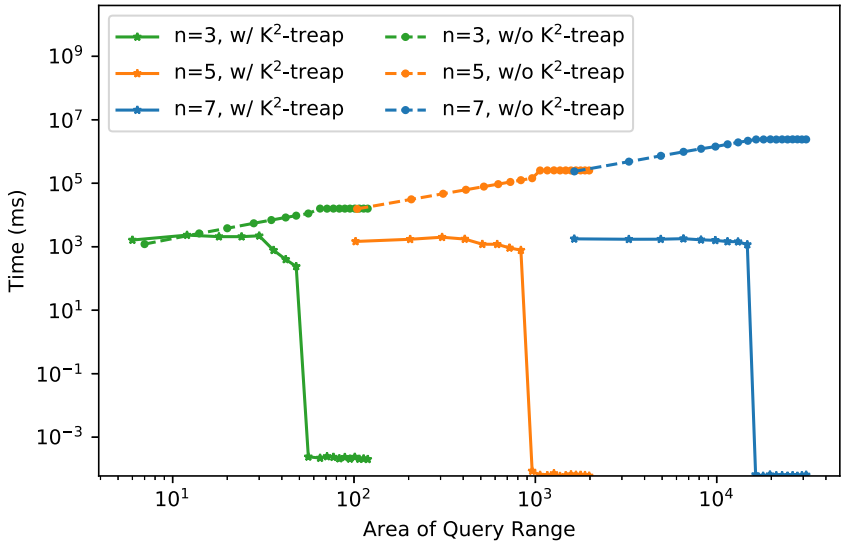


Fig. 6 Average time consumption for range-max queries of different query ranges. In this figure, the average query time of our proposed scheme decreases as the area of the query range increases, while that of the scheme without integrating the K^2-treap is linear to the query range's area

whole region, and none of them consider the range query in a specific region.

Some privacy-preserving range query schemes [23–26] have been proposed. The privacy-preserving range query is to obtain the data in a specific region while preserving the privacy of the query range and query result. Agrawal et al. [23] first presented the order-preserving encryption(OPE) technique that supports range queries over encrypted data. Boldyreva et al. [24] gave an efficient OPE implementation for a range query protocol. However, since the OPE technique essentially leaks the order information of the data, this scheme discloses the frequency of each different value in the dataset and is vulnerable to statistical attacks. Besides, this scheme cannot support the incremental update directly over ciphertexts. Lu [25] and Mahdikhani et al. [26] respectively proposed a new privacy-preserving range query scheme based on the homomorphic encryption technique. Both of the proposed schemes in [25, 26] focus on the scenario of the fog-enhanced IoT and can achieve range queries while preserving the privacy of IoT devices' plaintext data and users' query range. However, these schemes involve many interactions between the fog node and the IoT devices during the query process. Lu [25] reorganized range query by using range query expression, decomposition, and composition technique to achieve $O(\sqrt{n})$ communication. Mahdikhani et al. [26] proposed a novel range decomposition technique to compile the range query to achieve $O(\log^3 n)$ communication.

Different from the above existing schemes, we focus on the range-max query, and propose an efficient and privacy-preserving range-max query scheme in fog-based smart agriculture IoT, which can obtain the max/min data of any specific region (including the whole region). To protect the security of the proposed scheme, we design data encryption and comparison technique based on BGN homomorphic encryption technique. The adoption of the BGN homomorphic encryption technique not only strengthens the security of the proposed scheme but also enables the proposed scheme to support the incremental updates over ciphertexts. Meanwhile, our range-max query scheme is a non-interactive scheme, so there are no interactions between the IoT devices and the fog node during the query process.

8 Conclusion

In this paper, we have focused on the range-max query and proposed an efficient and privacy-preserving range-max query in fog-based smart agriculture IoT. Specifically, we first build the K^2-treap, which is used for storing the data collected by agriculture IoT devices and support efficient range-max query and dynamic update of the data. Then, we design a data encryption and comparison algorithm based on BGN homomorphic encryption technique and present an efficient and privacy-preserving range-max query in the fog-based smart agriculture IoT, which can not only securely compare two data based on their ciphertexts but also support the incremental update directly over ciphertexts. Notably, our comparison technique and range-max queries are run by the fog node, so there are no interactions between the agricultural IoT devices and the fog node during the comparison and query. Finally, we conduct a detailed security analysis and performance evaluation. The results show that our proposed scheme can indeed protect the privacy of the agricultural IoT devices' data and query results, and the experimental test results prove that our proposed scheme is efficient.

In future work, we will continue to focus on the privacy protection of smart agricultural IoT applications and further

explore some general and efficient privacy-preserving range query solutions.

Acknowledgements This work was supported by the China Scholarship Council when Min Zhou was visiting the University of New Brunswick, Canada. This work was also supported in part by the National Natural Science Foundation of China (No. 61872152, 61872409, 61902132), the Natural Science Foundation of Guangdong Province (No. 2018A03 0310147), Guangdong Basic and Applied Basic Research Foundation (No. 2019B030302008, No. 2020A1515010751), Science and Technology Program of Guangzhou (No.201902010081), Guangzhou Key Laboratory of Intelligent Agriculture.

References

1. Vasisht D, Kapetanovic Z, Won J, Jin X, Chandra R, Sinha SN, Kapoor A, Sudarshan M, Stratman S (2017) Farmbeats: An iot platform for data-driven agriculture. In: 14th USENIX Symposium on networked systems design and implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017, pp 515–529
2. Kamilaris A, Gao F, Prenafeta-Boldu FX, Ali MI (2016) Agri-iot: A semantic framework for internet of things-enabled smart farming applications. In: 3rd IEEE World Forum on internet of things, WF-IoT 2016, Reston, VA, USA, December 12-14, 2016, pp 442–447
3. Bonomi F, Milito RA, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC@SIGCOMM 2012, Helsinki, Finland, August 17, 2012, pp. 13–16
4. Bonomi F, Milito RA, Natarajan P, Zhu J (2014) Fog computing: A platform for internet of things and analytics. In: Big data and internet of things: A roadmap for smart environments, studies in computational intelligence, vol 546, pp 169–186
5. Chiang M, Zhang T (2016) Fog and iot: An overview of research opportunities. IEEE Int Things J 3(6):854–864
6. Dastjerdi AV, Buyya R (2016) Fog computing: Helping the internet of things realize its potential. IEEE Comput 49(8):112–116
7. Aazam M, Zeadally S, Harras KA (2018) Fog computing architecture, evaluation, and future research directions. IEEE Commun Mag 56(5):46–52
8. Yan Q, Yang H, Vuran MC, Irmak S (2017) SPRIDE: Scalable and private continual geo-distance evaluation for precision agriculture. In: 2017 IEEE Conference on communications and network security, CNS 2017, Las Vegas, NV, USA, October 9-11, 2017, pp 1–9
9. Huang C, Liu D, Ni J, Lu R, Shen X (2018) Reliable and privacy-preserving selective data aggregation for fog-based iot. In: 2018 IEEE International conference on communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018, pp 1–6
10. Lin X, Ni J, Shen XS (2018) Privacy-enhancing fog computing and its applications springer briefs in electrical and computer engineering
11. Xue L, Liu D, Huang C, Lin X, Shen XS (2020) Secure and privacy-preserving decision tree classification with lower complexity. J Commun Inf Networks 5(1):16–25
12. Shi X, An X, Zhao Q, Liu H, Xia L, Sun X, Guo Y (2019) State-of-the-art internet of things in protected agriculture. Sensors 19(8):1833
13. Ferrag MA, Shu L, Yang X, Derhab A, Maglaras LA (2020) Security and privacy for green iot-based agriculture: Review, blockchain solutions, and challenges. IEEE Access 8:32031–32053
14. Gupta M, Abdelsalam M, Khorsandroo S, Mittal S (2020) Security and privacy in smart farming: Challenges and opportunities. IEEE Access 8:34564–34584
15. Chen L, Lu R, Cao Z, Alharbi K, Lin X (2015) Muda: Multifunctional data aggregation in privacy-preserving smart grid communications. Peer Peer Netw Appl 8(5):777–792
16. Chen L, Lu R, Cao Z (2015) PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications. Peer Peer Netw Appl 8(6):1122–1132
17. Ge S, Zeng P, Lu R, Choo KR (2018) FGDA: Fine-grained data analysis in privacy-preserving smart grid communications. Peer Peer Netw Appl 11(5):966–978
18. Zheng Y, Lu R, Li B, Shao J, Yang H, Choo KR (2019) Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data. Inf Sci 498:91–105
19. Zheng Y, Lu R, Shao J (2019) Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data. J Medical Syst 43(5):123:1–123:13
20. Yao Y, Xiong N, Park JH, Ma L, Liu J (2013) Privacy-preserving max/min query in two-tiered wireless sensor networks. Comput Math Appl 65(9):1318–1325
21. Samanthula BK, Jiang W, Madria S (2013) A probabilistic encryption based MIN/MAX computation in wireless sensor networks. In: 2013 IEEE 14th International conference on mobile data management, Milan, Italy, June 3-6, 2013 - Volume 1, pp 77–86
22. Dai H, Ji Y, Xiao F, Yang G, Yi X, Chen L (2019) Privacy-preserving MAX/MIN query processing for WSN -as-a -service. In: 2019 IFIP networking conference, networking 2019, Warsaw, Poland, May 20-22, 2019, pp 1–9
23. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order-preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004, pp 563–574
24. Boldyreva A, Chenette N, Lee Y, O'Neill A (2009) Order-preserving symmetric encryption. In: Advances in Cryptology - EUROCRYPT 2009, 28th Annual international conference on the theory and applications of cryptographic techniques, Cologne, Germany, April 26-30, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5479, pp 224–241
25. Lu R (2019) A new communication-efficient privacy-preserving range query scheme in fog-enhanced iot. IEEE Int Things J 6(2):2497–2505
26. Mahdikhani H, Lu R, Zheng Y, Shao J, Ghorbani AA (2020) Achieving o($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot. IEEE Int Things J 7(6):5220–5232
27. Boneh D, Goh E, Nissim K (2005) Evaluating 2-dnf formulas on ciphertexts. In: Theory of cryptography, second theory of cryptography conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3378, pp 325–341
28. Lu R (2016) Privacy-Enhancing Aggregation Techniques for Smart Grid Communications Wireless Networks
29. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography

Min Zhou received the Ph.D. degree in information security from South China Agriculture University, Guangzhou, China, in 2013. She is an associate professor in the Department of Computer Science and Engineering, South China Agricultural University, Guangzhou, China. She is currently a visiting scholar at the Faculty of Computer Science, University of New Brunswick, Canada. Her research interests include IoT, network security, and privacy-preserving.

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. Her research interest includes cloud computing security, big data privacy and applied privacy.

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Limin Peng received the Ph.D. degree in Computer Science from South China University of Technology, Guangzhou, China, in 2011. Now he is working as an associate professor in Department of in Computer Science and Engineering, South China Agricultural University, Guangzhou, China. His research interests include data center networking, and cloud computing.

Rongxing Lu (S'09-M'11-SM'15) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. He is presently a senior member of IEEE Communications Society. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu currently serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Affiliations

Min Zhou[1,2] · Yandong Zheng[2] · Yunguo Guan[2] · Limin Peng[1] · Rongxing Lu[2]

Yandong Zheng
yzheng8@unb.ca

Yunguo Guan
yguan4@unb.ca

Limin Peng
penglm86@scau.edu.cn

Rongxing Lu
rlu1@unb.ca

[1] College of Mathematics and Informatics, South China Agriculture University, Guangzhou, 510642, China

[2] Faculty of Computer Science, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada