Secure and Efficient Multi-Attribute Range Queries based on Comparable Inner Product Encoding

Qin Liu[†*], SiXia Wu[†], Shuyu Pei[†], Jie Wu[‡], Tao Peng[§], and Guojun Wang[§]
[†]College of Computer Science and Electronic Engineering, Hunan University, P. R. China, 410082
[‡]Center for Networked Computing, Temple University, Philadelphia, PA 19122, USA
[§] School of Computer Science and Educational Software, Guangzhou University, P. R. China, 510006
[*]Correspondence to: gracelq628@hnu.edu.cn

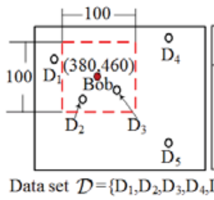
***Abstract*—Encryption, a powerful tool for data security, has been widely applied to protect sensitive data stored on untrusted cloud servers. One important problem in such an environment is how to support advanced query predicates, such as range queries, over an encrypted data set in an efficient and secure way. Order-preserving encryption (OPE) produces ciphertexts that preserve the order of their plaintexts and performs range queries directly on ciphertexts. However, *ideally secure* OPE schemes are inefficient (interactive and stateful), because they either ask for extensive client-to-server interactions or require a large persistent client storage that relates to the size of the data set. In this paper, we propose a comparable inner product encoding (CIPE) scheme to support *multi-attribute range queries* over encrypted data. Our main idea is to encode data and query values as encrypted vectors so that order comparison is realized by calculating the vector's inner product. Compared with existing OPE schemes, our scheme has the following merits: 1) *High efficiency.* It allows a client to retrieve data of interest in one round without maintaining any local state. 2) *Enhanced security.* It achieves ideal security while effectively resisting inference attacks that existing OPE schemes are vulnerable to. Extensive experiments conducted on a real-world, large-scale data set verify the effectiveness of our scheme.**

Keywords*—*Cloud computing, order-preserving encryption, multi-attribute range queries, security, efficiency.

I. Introduction

Nowadays, data creation is occurring at an alarming rate, which is overwhelming the data-processing capabilities and data storage capacities of traditional platforms. Cloud computing that enables ubiquitous and on-demand access to a shared pool of configurable computing resources has overwhelming advantages over traditional platforms. As an emerging trend, companies and organizations have begun to outsource their massive data sets to cloud servers to lessen the burden on local data storage improve the quality of service [1]–[3].

Application scenario. In order to offer better after-sales services, air-conditioning manufacturer X outsources customer data set $\mathcal{D}$ to Amazon EC2 so that its service engineers can search customers reported breakdowns in their physical proximity anytime and anywhere using mobile devices with geo-positioning capabilities (e.g., GPS). Assume that each data record $D_i \in \mathcal{D}$ that contains a specific customer's profile (e.g., name, contact information, and purchase date) is associated with two-dimensional geographic coordinates (x_i, y_i). To retrieve detailed information about customers within M meters, a service engineer located at (x', y') first generates a rectangle B, where the center is (x', y') and the edge length is $2M$. He then sends a range query $\mathcal{Q} = [b_l, b_u]$ to a cloud server,



Attribute	A_1 x-coordinate	A_2 y-coordinate
D_1	300	480
D_2	350	420
D_3	400	440
D_4	450	520
D_5	450	300

Description of $\mathcal{D}$

Fig. 1. Application scenario. The rectangle of dashed red edges is a query.

where $b_l = (x' - M, y' - M)$ and $b_u = (x' + M, y' + M)$ denote B's lower-left corner and upper-right corner, respectively. Once receiving $\mathcal{Q}$, the cloud server compares each data record's coordinates with b_l and b_u and returns those falling within B. For example, given a customer data set $\mathcal{D}$ as shown in Fig. 1, Bob located at $(380, 460)$ generates a query $\mathcal{Q} = [(330, 410), (430, 510)]$ to retrieve customer information within 50 meters, and the cloud server returns $\{D_2, D_3\}$.

Motivation. In the above scenario, the customer data is of high commercial values and is relevant to customers' privacy. To prevent unsolicited data access, sensitive data should be encrypted before outsourcing. However, data encryption makes evaluating various query predicates against ciphertextes a challenging problem. This is even harder for range queries where comparisons need to be performed based on ciphertexts.

Challenge. Order-preserving encryption (OPE) [4]–[9] produces ciphertexts that preserve the order of their plaintexts and allows order comparison on ciphertexts in the same way as plaintexts. While this makes OPE useful when performing range queries on ciphertexts, it also limits security. As shown in [10]–[12], almost of all existing OPE schemes are vulnerable to inference attacks even if they are *ideally secure* like [6]–[8]. Additionally, Lewi et.al [13] show that all ideal OPE schemes are inefficient (interactive and stateful); they either ask for extensive client-to-server interactions or require a large persistent client storage that relates to the size of the data set. Other techniques that support compute-then-compare are too expensive to implement, such as order-revealing encryption (ORE) [13]–[17] and homomorphic encryption [18], [19].

Our contributions. In this paper, we focus on supporting *multi-attribute range queries* over an encrypted data set, where each data record is described with many attributes, and for each attribute, a range query is generated to filter search results. To achieve ideal security in a non-interactive and stateless way, we propose a comparable inner product encoding (CIPE) scheme that encodes each attribute value a associated with a data record and each attribute value b in a query as two vectors $\mathbf{p}$ and $\mathbf{q}$, respectively, such that the inner product of $\mathbf{p}$ and $\mathbf{q}$, denoted as $\mathbf{p} \cdot \mathbf{q}$, has the same sign as that of $(b - a)$. Meanwhile, vectors will be encrypted with a secure k-nearest neighbor scheme (KNN for short) [20], thus, the order of two plaintexts can be quantified by judging the sign of the inner product of their encrypted vectors.

KNN with splitting and randomization mechanisms enables *index indistinguishability* and *trapdoor unlinkability*, but has failed to resist linear analyses [23]. Therefore, we first provide a basic CIPE construction, $CIPE_0$, which is secure in the known ciphertext model. Then, we provide an advanced construction, $CIPE_S$, to resist chosen plaintext attacks inherit from KNN. Compared with existing OPE schemes, our CIPE scheme has the following merits: 1) *High efficiency.* It allows a client to retrieve data of interest in one round without maintaining any local state. 2) *Enhanced security.* It can resist inference attacks that existing OPE schemes are vulnerable to. Our main contributions are summarized as follows:

- We propose a CIPE scheme to achieve efficient and secure multi-attribute range queries on encrypted data.
- Two constructions are provided to achieve ideal security in different security models.
- We evaluate the performance of our CIPE scheme on a real-world, large-scale data set. Experiment results demonstrate that our scheme is extremely efficient for secure range queries. At best, it needs only around 1.4 seconds on average while performing two-attribute range queries on 1 million encrypted data records.

II. Preliminaries

A. System Model

Our system model consists of three different parties: a data owner, a client, and cloud servers. The data owner possesses a proprietary data set $\mathcal{D}$, where each data record is described by multiple attributes. For data security, the data owner first builds a set of secure indexes $\mathcal{I}$ from $\mathcal{D}$, and then uploads the encrypted data set and the secure indexes $(\mathcal{C}, \mathcal{I})$ to a cloud server. In order to retrieve detailed information about data records matching query $\mathcal{Q}$, an authorized client first obtains the trapdoor $T_{\mathcal{Q}}$ (i.e., the encrypted query) from the data owner, then submits $T_{\mathcal{Q}}$ to the cloud server. The cloud server will evaluate $T_{\mathcal{Q}}$ over the secure indexes $\mathcal{I}$ and returns search results $\mathcal{C}_{\mathcal{Q}}$ to the client.

B. Adversary Model

The data owner and the client, collectively referred to as *cloud users*, are assumed to be fully trusted. The cloud server is the potential adversary and is assumed to be *honest but curious*. In other words, the cloud server always correctly executes a given protocol, but still attempts to learn additional information about the stored data and the received message. In terms of the information available to the cloud server, we mainly consider the following models used in existing work [20]:

Known ciphertext model. The cloud server only knows the encrypted data set, the secure indexes, the submitted trapdoors, and the returned search results.

Known background model. The cloud server can know additional background information besides that in the known ciphertext model. The background refers to the keyword, frequency and the distribution among the file collection, which can be used to infer certain (plaintext, ciphertext) pairs.

While data privacy can be preserved through standard symmetric encryption, e.g., AES, our privacy objective is to protect cloud users' privacy in the following aspects:

- **Attribute secrecy.** The cloud server cannot deduce attribute values from either secure indexes or trapdoors.
- **Index indistinguishability and trapdoor unlinkability.** Given a set of secure indexes (or a set of trapdoors), the cloud server cannot decide whether they are generated for the same attribute value or not.

C. Design Goals

Our design goals are to offer advanced search services in cloud computing by simultaneously achieving the following:

- **High efficiency**. The client can retrieve data of interests in one round without keeping any local storage.
- **High scalability**. Multi-attribute range queries can be effectively performed on a large-scale data set.
- **Enhanced security**. With ideal security, the proposed scheme can effectively resist inference attacks.
- **Enhanced privacy**. Our privacy objectives are preserved under chosen plaintext attacks.

D. Secure k-Nearest Neighbor Scheme

Given a vector $\mathbf{v}$, its i-th element is denoted by $\mathbf{v}[i]$. Let notations "$\star$" and "$\cdot$" denote matrix multiplication and vector inner product, and let $\mathbf{M}^{-1}$ and $\mathbf{M}^T$ denote the inverse and transposed matrices of $\mathbf{M}$, respectively. KNN tailored for our CIPE scheme mainly consists of the following algorithms:

• $GenKey(1^\kappa) \rightarrow sk$: It takes a security parameter $\kappa \in \mathbb{N}$ as input and generates the secret key $sk = (\mathbf{M}_1, \mathbf{M}_2, S)$, where $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{d \times d}$ are invertible matrices and S is a bit string of d bits. Note that $d \geq \kappa$ is large enough to prevent brute force attacks (e.g., $d = \kappa = 128$) and that the number of 0s is approximately equal to the number of 1s in S.

• $EncI(\mathbf{p}, sk) \rightarrow \mathbf{p}'$: It splits index vector $\mathbf{p} \in \mathbb{R}^d$ into two vectors, $(\mathbf{p}_{|\alpha|}, \mathbf{p}_{|\beta|})$: for $i = 1$ to d, if the i-th bit of S equals 1, $\mathbf{p}[i]$ is randomly split into $\mathbf{p}_{|\alpha|}[i]$ and $\mathbf{p}_{|\beta|}[i]$ such that $\mathbf{p}_{|\alpha|}[i] + \mathbf{p}_{|\beta|}[i] = \mathbf{p}[i]$; otherwise, both $\mathbf{p}_{|\alpha|}[i]$ and $\mathbf{p}_{|\beta|}[i]$ are set to $\mathbf{p}[i]$. The encrypted index is a pair of vectors $\mathbf{p}' = (\mathbf{p}'_{|\alpha|}, \mathbf{p}'_{|\beta|})$ where $(\mathbf{p}'_{|\alpha|} = \mathbf{M}_1^T \star \mathbf{p}_{|\alpha|}, \mathbf{p}'_{|\beta|} = \mathbf{M}_2^T \star \mathbf{p}_{|\beta|})$.

• $EncQ(\mathbf{q}, sk) \rightarrow \mathbf{q}'$: It splits query vector $\mathbf{q} \in \mathbb{R}^d$ into two vectors, $(\mathbf{q}_{|\alpha|}, \mathbf{q}_{|\beta|})$: for $i = 1$ to d, if the i-th bit of S equals 0, $\mathbf{q}[i]$ is split into $\mathbf{q}_{|\alpha|}[i]$ and $\mathbf{q}_{|\beta|}[i]$ such that $\mathbf{q}[i] = \mathbf{q}_{|\alpha|}[i] + \mathbf{q}_{|\beta|}[i]$; otherwise, both $\mathbf{q}_{|\alpha|}[i]$ and $\mathbf{q}_{|\beta|}[i]$ are set to $\mathbf{q}[i]$. The encrypted query is a pair of vectors $\mathbf{q}' = (\mathbf{q}'_{|\alpha|}, \mathbf{q}'_{|\beta|})$ where $(\mathbf{q}'_{|\alpha|} = \mathbf{M}_1^{-1} \star \mathbf{q}_{|\alpha|}, \mathbf{q}'_{|\beta|} = \mathbf{M}_2^{-1} \star \mathbf{q}_{|\beta|})$.

• $Cal(\mathbf{p}', \mathbf{q}') \rightarrow v$: It calculates $v = \mathbf{p}'_{|\alpha|} \cdot \mathbf{q}'_{|\alpha|} + \mathbf{p}'_{|\beta|} \cdot \mathbf{q}'_{|\beta|}$ as the result. Note that the output of the $KNN.Cal$ algorithm equals the inner product of $\mathbf{p}$ and $\mathbf{q}$:

$$\begin{aligned} & \mathbf{p}'_{|\alpha|} \cdot \mathbf{q}'_{|\alpha|} + \mathbf{p}'_{|\beta|} \cdot \mathbf{q}'_{|\beta|} \\ &= (\mathbf{M}_1^T \star \mathbf{p}_{|\alpha|}) \cdot (\mathbf{M}_1^{-1} \star \mathbf{q}_{|\alpha|}) + (\mathbf{M}_2^T \star \mathbf{p}_{|\beta|}) \cdot (\mathbf{M}_2^{-1} \star \mathbf{q}_{|\beta|}) \\ &= \mathbf{p}_{|\alpha|}^T \star \mathbf{q}_{|\alpha|} + \mathbf{p}_{|\beta|}^T \star \mathbf{q}_{|\beta|} = \mathbf{p} \cdot \mathbf{q} \end{aligned}$$

TABLE I. SUMMARY OF NOTATIONS

$\mathcal{D}$	A set of n data records $\{D_1, \ldots, D_n\}$
$\mathcal{A}_i$	A set of m attribute values $(a_{i_1}, \ldots, a_{i_m})$ of D_i
$\mathcal{I}_i$	A multi-attribute secure index $(I_{i_1}, \ldots, I_{i_m})$ of D_i
$\mathcal{Q}_j$	A multi-attribute range query $(Q_{j_1}, \ldots, Q_{j_m})$
Q_{j_k}	A range query $[b_{j_{k,l}}, b_{j_{k,u}}]$ on the k-th attribute in query $\mathcal{Q}_j$
$T_{\mathcal{Q}_j}$	A multi-attribute trapdoor $(T_{Q_{j_1}}, \ldots, T_{Q_{j_m}})$ of $\mathcal{Q}_j$
$\mathbf{p}_{i_k}$	An index vector of the k-th attribute's value a_{i_k}
$\mathbf{q}_{j_{k,\phi}}, \mathbf{Q}_{j_{k,\phi}}$	A query vector/matrix of $b_{j_{k,\phi}}$ for $\phi \in \{l, u\}$

III. OVERVIEW

A. Notations and Definitions

For $x, y \in \mathbb{N}$ and $x < y$, we write notations $[x]$ and $[x \sim y]$ to represent the set of integers in $\{1, \ldots, x\}$ and $\{x, \ldots, y\}$, respectively. The most relevant notations are shown in Table I.

Attribute. The data set $\mathcal{D} = \{D_1, \ldots, D_n\}$ is described by m attributes $(A_1, \ldots, A_m)$, where each attribute A_k takes its *numerical* value from a specified domain $\mathbb{D}_k$ for $k \in [m]$. Therefore, each data record $D_i \in \mathcal{D}$ is actually associated with m attribute values $\mathcal{A}_i = (a_{i_1}, \ldots, a_{i_m})$ based on which a multi-attribute secure index $\mathcal{I}_i = (I_{i_1}, \ldots, I_{i_m})$ is built, where I_{i_k} is a secure index on attribute value a_{i_k} for $k \in [m]$. The primary role of attributes is making comparisons at the time of query. Therefore, there is no need to use authentic attribute values. In our CIPE scheme, all attribute values can be preprocessed by order preserving functions like [21].

Query. We mainly consider a multi-attribute range query $\mathcal{Q}_j = (Q_{j_1}, \ldots, Q_{j_m})$, where $Q_{j_k} = [b_{j_{k,l}}, b_{j_{k,u}}]$ is a range query on attribute A_k with the lower bound $b_{j_{k,l}}$ and the upper bound $b_{j_{k,u}}$ for $k \in [m]$. Given $\mathcal{Q}_j$, a multi-attribute trapdoor $T_{\mathcal{Q}_j} = (T_{Q_{j_1}}, \ldots, T_{Q_{j_m}})$ is created where $T_{Q_{j_k}}$ is a trapdoor of Q_{j_k} for $k \in [m]$. The boolean formulas connecting different dimensions are "AND" operators. That is, each range query in $\mathcal{Q}_j$ further narrows search results. We define a data record D_i matching a query $\mathcal{Q}_j$, denoted as $D_i \bowtie \mathcal{Q}_j$, as follows:

Definition 1 (Matching)**.** *Given D_i associated with $(a_{i_1}, \ldots, a_{i_m})$ and $\mathcal{Q}_j = (Q_{j_1}, \ldots, Q_{j_m})$, where $Q_{j_k} = [b_{j_{k,l}}, b_{j_{k,u}}]$, we have $D_i \bowtie \mathcal{Q}_j$ if for $k \in [m]$, $a_{i_k} \geq b_{j_{k,l}}$ and $a_{i_k} \leq b_{j_{k,u}}$.*

B. The Definition of CIPE

A CIPE scheme consists of the following algorithms:

• $KeyGen(1^\kappa) \rightarrow SK$: The data owner takes the security parameter κ as the input and outputs a secret key SK.

• $SecureInx(\mathcal{A}_i, SK) \rightarrow \mathcal{I}_i$: Given a set of attribute values $\mathcal{A}_i$, the data owner builds a multi-attribute secure index $\mathcal{I}_i$ for data record D_i with secret key SK. The collection of secure indexes built for data set $\mathcal{D}$ is set to $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$.

• $Trapdoor(\mathcal{Q}_j, SK) \rightarrow T_{\mathcal{Q}_j}$: Given a multi-attribute range query $\mathcal{Q}_j$, the data owner builds a multi-attribute trapdoor $T_{\mathcal{Q}_j}$ with secret key SK.

• $Search(\mathcal{I}_i, T_{\mathcal{Q}_j}) \rightarrow \{0, 1\}$: The cloud server evaluates the trapdoor $T_{\mathcal{Q}_j}$ on the secure index $\mathcal{I}_i$, and outputs 1 if $D_i \bowtie \mathcal{Q}_j$. Otherwise, it outputs 0.

Definition 2 (Correctness of CIPE)**.** *Given a secret key SK generated by the $KeyGen$ algorithm, a secure index $\mathcal{I}_i$ output by the $SecureInx$ algorithm, and a trapdoor $T_{\mathcal{Q}_j}$ output by the $Trapdoor$ algorithm, our CIPE scheme is correct if the $Search(\mathcal{I}_i, T_{\mathcal{Q}_j})$ algorithm outputs 1 if $D_i \bowtie \mathcal{Q}_j$.*

C. Security Definition

As in [13], [17], we utilize a leakage function $\mathcal{L}$ to capture what is being revealed during the search process. Let $\kappa \in \mathbb{N}$ be the security parameter, we consider a simulation-based game between a probabilistic polynomial-time (PPT) simulator Sim and a PPT adversary Adv as follows:

• $\mathbf{REAL}_{\mathrm{Adv}}(\kappa)$: The challenger runs $KeyGen(1^\kappa)$ to generate secret key SK. Adversary Adv outputs $\mathcal{D}$ and receives $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ from the challenger so that for $i \in [n]$, $\mathcal{I}_i \leftarrow SecureInx(\mathcal{A}_i, SK)$. Adv makes a polynomial number of adaptive queries $\mathbb{Q} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_q)$ and for each query $\mathcal{Q}_j \in \mathbb{Q}$, Adv receives a trapdoor $T_{\mathcal{Q}_j}$ from the challenger such that $T_{\mathcal{Q}_j} \leftarrow Trapdoor(\mathcal{Q}_j, SK)$. Finally, Adv outputs $V = (\mathcal{I}, \mathbb{T})$, where $\mathbb{T} = (T_{\mathcal{Q}_1} \ldots, T_{\mathcal{Q}_q})$.

• $\mathbf{SIM}_{\mathrm{Adv},\mathrm{Sim},\mathcal{L}}(\kappa)$: Adversary Adv outputs $\mathcal{D}$. Given $\mathcal{D}$, simulator Sim generates and sends $\mathcal{I}$ to Adv. Adv makes a polynomial number of adaptive queries $\mathbb{Q} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_q)$. Given a leakage function $\mathcal{L}$, Sim returns an appropriate trapdoor $T_{\mathcal{Q}_j}$ for each query $\mathcal{Q}_j \in \mathbb{Q}$. Finally, Adv outputs $V = (\mathcal{I}, \mathbb{T})$, where $\mathbb{T} = (T_{\mathcal{Q}_1} \ldots, T_{\mathcal{Q}_q})$.

Definition 3 (Security of CIPE)**.** *Our CIPE scheme is secure with leakage function $\mathcal{L}$ if for all PPT adversaries Adv, there is a PPT simulator Sim such that $|\Pr[\mathbf{REAL}_{\mathrm{Adv}}(\kappa) = 1] - \Pr[\mathbf{SIM}_{\mathrm{Adv},\mathrm{Sim},\mathcal{L}}(\kappa) = 1]|$ is negligible.*

As defined in [4], indistinguishability under an ordered chosen attack (IND-OCPA), the ideal security of OPE, intuitively says that ciphertexts must not leak anything besides orders of the plaintexts. Therefore, the best-possible notion of simulation-security is secure with respect to leakage function $\mathcal{L} = \{(i, j, Search(\mathcal{I}_i, T_{\mathcal{Q}_j}) | i \in [n], j \in [q]\}$. The outputs of the $Search$ algorithm reveal order of plaintexts.

IV. COMPARABLE INNER PRODUCT ENCODING (CIPE)

A. Main Idea

Our CIPE scheme is built on top of the index vector encoding algorithm (Alg. 1) and the query vector encoding algorithm (Alg. 2). The main idea of both algorithms is to encode a data value a and a query value b as vectors, so that the sign of vector inner product is the same as that of $(b - a)$, and it can be used to determine whether a data record matches a query or not. In a single-attribute data set (the subscript k in $a_{i_k}, b_{j_{k,\phi}}, Q_{j_k}, \mathbf{p}_{i_k}$, and $\mathbf{q}_{j_{k,\phi}}$ for $\phi \in \{l, u\}$ is omitted), vector $\mathbf{p}_i$ is constructed for attribute value a_i in data record D_i and given a range query $Q_j = [b_{j_l}, b_{j_u}]$, vectors $\mathbf{q}_{j_l}$ and $\mathbf{q}_{j_u}$ are constructed for b_{j_l} and b_{j_u}, respectively; we have $D_i \bowtie \mathcal{Q}_j$ if $\mathbf{p}_i \cdot \mathbf{q}_{j_l} \leq 0$ and $\mathbf{p}_i \cdot \mathbf{q}_{j_u} \geq 0$. When extended to a data set with m attributes, vector $\mathbf{p}_{i_k}$ is constructed for the k-th attribute's value a_{i_k} in data record D_i, and given a range query $Q_{j_k} = [b_{j_{k,l}}, b_{j_{k,u}}]$ on the k-th attribute, vectors $\mathbf{q}_{j_{k,l}}$ and $\mathbf{q}_{j_{k,u}}$ are constructed for $b_{j_{k,l}}$ and $b_{j_{k,u}}$, respectively. We have $D_i \bowtie \mathcal{Q}_j$ if $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,l}} \leq 0$ and $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,u}} \geq 0$ for $k \in [m]$.

As the simplest construction for a single-attribute data set, given an even number d, a random bit string L of length d (where the number of 0s equals the number of 1s) is generated. For attribute value a_i in data record D_i, a d-dimensional vector $\mathbf{p}_i$ is constructed such that $\mathbf{p}_i[\iota] = a_i$ if the ι-th bit of L is 1 and $\mathbf{p}_i[\iota] = 1$ if the ι-th bit of L is 0. Given a range query $Q_j = [b_{j_l}, b_{j_u}]$, two d-dimensional vectors $\mathbf{q}_{j_l}$ and $\mathbf{q}_{j_u}$ are constructed such that $\mathbf{q}_{j_l}[\iota] = \mathbf{q}_{j_u}[\iota] = -1$ if the ι-th bit of

Algorithm 1 Index Vector Encoding

```
Require: attribute value $a_{i_k}$ associated with data record $D_i$ for $i \in [n]$ and $k \in [m]$, two bit strings $L_1$ and $L_2$ of $\frac{d}{2}$ bits, and a shuffle function $\sigma$
Ensure: index vector $\mathbf{p}_{i_k} \in \mathbb{R}^d$
 1: for $\iota \in [\frac{d}{2}]$ do
 2:     if the $\iota$-th bit of $L_1$ equals 1 then
 3:         set $\mathbf{p}_{i_k}[\iota] = a_{i_k}$
 4:     else
 5:         set $\mathbf{p}_{i_k}[\iota] = 1$
 6: generate a vector $\mathbf{C} = (c_1, \dots, c_{\frac{d}{4}})$ where $c_\iota \in \mathbf{C}$ is randomly chosen from $\mathbb{R}^+ \cup \{0\}$ and $\sum_{\iota=1}^{\frac{d}{4}} c_\iota > 0$.
 7: set $x = y = 1$
 8: for $\iota \in [\frac{d}{2} + 1 \sim d]$ do
 9:     if the $(\iota - \frac{d}{2})$-th bit of $L_2$ equals 1 then
10:         set $\mathbf{p}_{i_k}[\iota] = a_{i_k} c_x$ and $x = x + 1$
11:     else
12:         set $\mathbf{p}_{i_k}[\iota] = c_y$ and $y = y + 1$
13: shuffle elements in $\mathbf{p}_{i_k}$ with $\sigma$
```

L is 1; $\mathbf{q}_{j_l}[\iota] = b_{j_l}$ and $\mathbf{q}_{j_u}[\iota] = b_{j_u}$ if the ι-th bit of L is 0. Since $\mathbf{p}_i \cdot \mathbf{q}_{j_l} = \frac{d}{2}(b_{j_l} - a_i)$ and $\mathbf{p}_i \cdot \mathbf{q}_{j_u} = \frac{d}{2}(b_{j_u} - a_i)$, we have $D_i \bowtie \mathcal{Q}_j$ if $\mathbf{p}_i \cdot \mathbf{q}_{j_l} \leq 0$ and $\mathbf{p}_i \cdot \mathbf{q}_{j_u} \geq 0$. For example, given $d = 4$ and $L = (1100)$, for D_1 and D_2 with attributes $a_1 = 3$ and $a_2 = 6$, we have $\mathbf{p}_1 = (3, 3, 1, 1)$ and $\mathbf{p}_2 = (6, 6, 1, 1)$, respectively. For $Q_1 = [2, 5]$, we have $\mathbf{q}_{1_l} = (-1, -1, 2, 2)$ and $\mathbf{q}_{1_u} = (-1, -1, 5, 5)$. Hence, we have $\mathbf{p}_1 \cdot \mathbf{q}_{1_l} = -2$, $\mathbf{p}_1 \cdot \mathbf{q}_{1_u} = 4$, $\mathbf{p}_2 \cdot \mathbf{q}_{1_l} = -8$, and $\mathbf{p}_2 \cdot \mathbf{q}_{1_u} = -2$. The results are in accordance with the fact that $D_1 \bowtie \mathcal{Q}_1$ and $D_2 \not\bowtie \mathcal{Q}_1$.

Although all vectors are encrypted with KNN for attribute secrecy, the simplest construction above will leak the following information besides order: 1) *The distance between attribute values*. The result of the inner product is multiples of the distance between attribute values. The larger the distance, the larger the result. 2) *The equality of attribute values*. Given two data records with the same attribute value, results of vector inner products will be equal. With such information, the adversary may infer the similarities between two data records. To hide such sensitive information, we multiply each inner product with a *random positive number* sampled from $\mathbb{R}^+$.

B. Basic Construction

Given a security parameter $\kappa \in \mathbb{N}$, and a secure KNN scheme $KNN = (GenKey, EncI, EncQ, Cal)$ as described in Section II-D, our basic CIPE construction is as follows:

• $KeyGen_0(1^\kappa) \rightarrow SK$: The data owner runs $KNN.GenKey(1^\kappa)$ to generate sk. Assume that $\eta \in \mathbb{N}$ is large enough so that $d = 2^{\eta+2} \geq \kappa$. She then randomly chooses two bit strings L_1 and L_2 of $\frac{d}{2}$ bits, and in both strings, the number of 0s equals the number of 1s. She chooses a permutation σ on $[d]$, and sets the secret key as $SK = (sk, \sigma, L_1, L_2)$.

• $SecureInx_0(\mathcal{A}_i, SK) \rightarrow \mathcal{I}_i$: For attribute value a_{i_k} associated with data record $D_i \in \mathcal{D}$ where $k \in [m]$, the data owner constructs a vector $\mathbf{p}_{i_k} \in \mathbb{R}^d$ by running Alg. 1. To generate encrypted vectors, she runs $KNN.EncI(\mathbf{p}_{i_k}, sk)$ to output $\mathbf{p}'_{i_k} = (\mathbf{p}'_{|\alpha|i_k}, \mathbf{p}'_{|\beta|i_k})$ for $k \in [m]$. Finally, she sets $\mathcal{I}_i = (I_{i_1}, \dots, I_{i_m})$, where $I_{i_k} = \mathbf{p}'_{i_k}$ for $k \in [m]$.

Algorithm 2 Query Vector Encoding

```
Require: attribute value $b_{j_{k,\phi}}$ associated with query $\mathcal{Q}_j$ for $k \in [m]$ and $\phi \in \{l, u\}$, two bit strings $L_1$ and $L_2$ of $\frac{d}{2}$ bits, and a shuffle function $\sigma$
Ensure: query vector $\mathbf{q}_{j_{k,\phi}} \in \mathbb{R}^d$
 1: generate a vector $\overline{\mathbf{C}} = (\overline{c}_1, \dots, \overline{c}_{\frac{d}{4}})$ where $\overline{c}_\iota \in \mathbf{C}$ is randomly chosen from $\mathbb{R}^+ \cup \{0\}$ and $\sum_{\iota=1}^{\frac{d}{4}} \overline{c}_\iota > 0$.
 2: set $x = y = 1$
 3: for $\iota \in [\frac{d}{2}]$ do
 4:     if the $\iota$-th bit of $L_1$ equals 1 then
 5:         set $\mathbf{q}_{j_{k,\phi}}[\iota] = -\overline{c}_x$ and $x = x + 1$
 6:     else
 7:         set $\mathbf{q}_{j_{k,\phi}}[\iota] = b_{j_{k,\phi}} \overline{c}_y$ and $y = y + 1$
 8: for $\iota \in [\frac{d}{2} + 1 \sim d]$ do
 9:     if the $(\iota - \frac{d}{2})$-th bit of $L_2$ equals 1 then
10:         set $\mathbf{q}_{j_{k,\phi}}[\iota] = -1$
11:     else
12:         set $\mathbf{q}_{j_{k,\phi}}[\iota] = b_{j_{k,\phi}}$
13: shuffle elements in $\mathbf{q}_{j_{k,\phi}}$ with $\sigma$
```

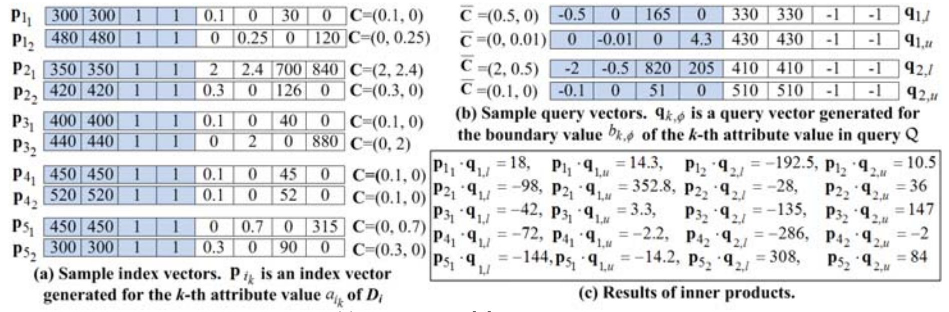
• $Trapdoor_0(\mathcal{Q}_j, SK) \rightarrow T_{\mathcal{Q}_j}$: Given range query $Q_{j_k} = [b_{j_{k,l}}, b_{j_{k,u}}]$ where $k \in [m]$, the data owner first constructs vectors $\mathbf{q}_{j_{k,l}}, \mathbf{q}_{j_{k,u}} \in \mathbb{R}^d$ for the lower bound $b_{j_{k,l}}$ and the upper bound $b_{j_{k,u}}$, respectively, by running Alg. 2. To generate encrypted vectors, she runs algorithm $KNN.EncQ$ to output $\mathbf{q}'_{j_{k,l}} = (\mathbf{q}'_{|\alpha|j_{k,l}}, \mathbf{q}'_{|\beta|j_{k,l}})$ and $\mathbf{q}'_{j_{k,u}} = (\mathbf{q}'_{|\alpha|j_{k,u}}, \mathbf{q}'_{|\beta|j_{k,u}})$, respectively. Finally, she sets $T_{\mathcal{Q}_j} = (T_{Q_{j_1}}, \dots, T_{Q_{j_m}})$ where $T_{Q_{j_k}} = (\mathbf{q}'_{j_{k,l}}, \mathbf{q}'_{j_{k,u}})$ for $k \in [m]$.

• $Search_0(\mathcal{I}_i, T_{\mathcal{Q}_j}) \rightarrow \{0, 1\}$: For $k \in [m]$, the cloud server runs $v_{k,l} \leftarrow KNN.Cal(\mathbf{p}'_{i_k}, \mathbf{q}'_{j_{k,l}})$ and $v_{k,u} \leftarrow KNN.Cal(\mathbf{p}'_{i_k}, \mathbf{q}'_{j_{k,u}})$ to calculate the inner products of encrypted vectors. It outputs 1 if $v_{k,l} \leq 0$ and $v_{k,u} \geq 0$ for $k \in [m]$. Otherwise, it outputs 0.

Example 1. Consider the application scenario shown in Fig. 1, where Bob issues query $\mathcal{Q} = ([b_{1,l}, b_{1,u}], [b_{2,l}, b_{2,u}]) = ([330, 430], [410, 510])$ to retrieve matched data records in $\mathcal{D} = \{D_1, \dots, D_5\}$. Sample index/query vectors are shown in Fig. 2-(a)2-(b). For example, $\mathbf{p}_{1_1}$ encoding attribute value 300 for D_1 is constructed as follows: Given $L_1 =$ "1100", we set $\mathbf{p}_{1_1}[1] = \mathbf{p}_{1_1}[2] = 300$ and $\mathbf{p}_{1_1}[3] = \mathbf{p}_{1_1}[4] = 1$. Then, we generates a vector $\mathbf{C} = (c_1, c_2) = (0.1, 0)$. Given $L_2 =$ "0011", we set $\mathbf{p}_{1_1}[5] = c_1 = 0.1$, $\mathbf{p}_{1_1}[6] = c_2 = 0$, $\mathbf{p}_{1_1}[7] = c_1 \times 300 = 30$, and $\mathbf{p}_{1_1}[8] = c_2 \times 300 = 0$. The query vectors are constructed in a similar way as the index vectors except that bit strings have the opposite effect and randomness are in the first half part of a query vector. The results in Fig. 2-(c) show that $\{D_2, D_3\}$ satisfies the query condition.

C. Correctness Analysis

Since permutation has no influence on the result of the inner product, we simply assume that $\sigma(\iota) \rightarrow \iota$ for $\iota \in [d]$. For $\iota \in [\frac{d}{2}]$, if the ι-th bit of L_1 is 1, we have $\mathbf{p}_{i_k}[\iota] = a_{i_k}$ and $\mathbf{q}_{j_{k,\phi}}[\iota] = -\overline{c}_x$; otherwise, we have $\mathbf{p}_{i_k}[\iota] = 1$ and $\mathbf{q}_{j_{k,\phi}}[\iota] = b_{j_{k,\phi}} \overline{c}_x$. Let vectors $\mathbf{v}_1 = (\mathbf{p}_{i_k}[1], \dots, \mathbf{p}_{i_k}[\frac{d}{2}])$ and $\mathbf{w}_1 = (\mathbf{q}_{j_{k,\phi}}[1], \dots, \mathbf{q}_{j_{k,\phi}}[\frac{d}{2}])$ denote the first half part of elements in $\mathbf{p}_{i_k}$ and $\mathbf{q}_{j_{k,\phi}}$, respectively. Since $-\overline{c}_x$ and $\overline{c}_x b_{j_{k,l}}$ come in pairs (lines 3-7 in Alg. 2), we have $\mathbf{v}_1 \cdot \mathbf{w}_1 =$



$\mathbf{p}_{1_1}$	300	300	1	1	0.1	0	30	0	C=(0.1, 0)
$\mathbf{p}_{1_2}$	480	480	1	1	0	0.25	0	120	C=(0, 0.25)
$\mathbf{p}_{2_1}$	350	350	1	1	2	2.4	700	840	C=(2, 2.4)
$\mathbf{p}_{2_2}$	420	420	1	1	0.3	0	126	0	C=(0.3, 0)
$\mathbf{p}_{3_1}$	400	400	1	1	0.1	0	40	0	C=(0.1, 0)
$\mathbf{p}_{3_2}$	440	440	1	1	0	2	0	880	C=(0, 2)
$\mathbf{p}_{4_1}$	450	450	1	1	0.1	0	45	0	C=(0.1, 0)
$\mathbf{p}_{4_2}$	520	520	1	1	0.1	0	52	0	C=(0.1, 0)
$\mathbf{p}_{5_1}$	450	450	1	1	0	0.7	0	315	C=(0, 0.7)
$\mathbf{p}_{5_2}$	300	300	1	1	0.3	0	90	0	C=(0.3, 0)

(a) Sample index vectors. $\mathbf{p}_{i_k}$ is an index vector generated for the *k*-th attribute value a_{i_k} of D_i

$\overline{\mathbf{C}}$ =(0.5, 0)	-0.5	0	165	0	330	330	-1	-1	$\mathbf{q}_{1,l}$
$\overline{\mathbf{C}}$ =(0, 0.01)	0	-0.01	0	4.3	430	430	-1	-1	$\mathbf{q}_{1,u}$
$\overline{\mathbf{C}}$ =(2, 0.5)	-2	-0.5	820	205	410	410	-1	-1	$\mathbf{q}_{2,l}$
$\overline{\mathbf{C}}$ =(0.1, 0)	-0.1	0	51	0	510	510	-1	-1	$\mathbf{q}_{2,u}$

(b) Sample query vectors. $\mathbf{q}_{k,\phi}$ is a query vector generated for the boundary value $b_{k,\phi}$ of the *k*-th attribute value in query Q

$\mathbf{p}_{1_1}\cdot\mathbf{q}_{1,l}=18$,	$\mathbf{p}_{1_1}\cdot\mathbf{q}_{1,u}=14.3$,	$\mathbf{p}_{1_2}\cdot\mathbf{q}_{2,l}=-192.5$,	$\mathbf{p}_{1_2}\cdot\mathbf{q}_{2,u}=10.5$
$\mathbf{p}_{2_1}\cdot\mathbf{q}_{1,l}=-98$,	$\mathbf{p}_{2_1}\cdot\mathbf{q}_{1,u}=352.8$,	$\mathbf{p}_{2_2}\cdot\mathbf{q}_{2,l}=-28$,	$\mathbf{p}_{2_2}\cdot\mathbf{q}_{2,u}=36$
$\mathbf{p}_{3_1}\cdot\mathbf{q}_{1,l}=-42$,	$\mathbf{p}_{3_1}\cdot\mathbf{q}_{1,u}=3.3$,	$\mathbf{p}_{3_2}\cdot\mathbf{q}_{2,l}=-135$,	$\mathbf{p}_{3_2}\cdot\mathbf{q}_{2,u}=147$
$\mathbf{p}_{4_1}\cdot\mathbf{q}_{1,l}=-72$,	$\mathbf{p}_{4_1}\cdot\mathbf{q}_{1,u}=-2.2$,	$\mathbf{p}_{4_2}\cdot\mathbf{q}_{2,l}=-286$,	$\mathbf{p}_{4_2}\cdot\mathbf{q}_{2,u}=-2$
$\mathbf{p}_{5_1}\cdot\mathbf{q}_{1,l}=-144$,	$\mathbf{p}_{5_1}\cdot\mathbf{q}_{1,u}=-14.2$,	$\mathbf{p}_{5_2}\cdot\mathbf{q}_{2,l}=308$,	$\mathbf{p}_{5_2}\cdot\mathbf{q}_{2,u}=84$

(c) Results of inner products.

Fig. 2. Working process of $CIPE_0$. Suppose that $d = 8$, $\sigma(\iota) \to \iota$ for $\iota \in [d]$, and bit strings $L_1 =$ "1100" and $L_2 =$ "0011".

$(\sum_{x=1}^{\frac{d}{4}} \bar{c}_x)(b_{j_{k,\phi}} - a_{i_k})$. For $\iota \in [\frac{d}{2}+1 \sim d]$, if the $(\iota - \frac{d}{2})$-th bit of L_2 is 1, we have $\mathbf{p}_{i_k}[\iota] = a_{i_k}c_x$ and $\mathbf{q}_{j_{k,\phi}}[\iota] = -1$; otherwise, we have $\mathbf{p}_{i_k}[\iota] = c_x$ and $\mathbf{q}_{j_{k,\phi}}[\iota] = b_{j_{k,\phi}}$.

Let vectors $\mathbf{v}_2 = (\mathbf{p}_{i_k}[\frac{d}{2}+1], \ldots, \mathbf{p}_{i_k}[d])$ and $\mathbf{w}_2 = (\mathbf{q}_{j_{k,\phi}}[\frac{d}{2}+1], \ldots, \mathbf{q}_{j_{k,\phi}}[d])$ denote the second half part of elements in $\mathbf{p}_{i_k}$ and $\mathbf{q}_{j_{k,\phi}}$, respectively. Similarly, c_x and $a_{i_k}c_x$ come in pairs (lines 8-12 in Alg. 1). Therefore, we have $\mathbf{v}_2 \cdot \mathbf{w}_2 = (\sum_{x=1}^{\frac{d}{4}} c_x)(b_{j_{k,\phi}} - a_{i_k})$ and $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,\phi}} = (\xi_{i_k} + \varepsilon_{j_{k,\phi}})(b_{j_{k,\phi}} - a_{i_k})$, where $\xi_{i_k} = \sum_{x=1}^{\frac{d}{4}} c_x$ and $\varepsilon_{j_{k,\phi}} = \sum_{x=1}^{\frac{d}{4}} \bar{c}_x$. In our construction, $(c_1, \ldots, c_{\frac{d}{4}})$ and $(\bar{c}_1, \ldots, \bar{c}_{\frac{d}{4}})$ are random numbers from $\mathbb{R}^+ \cup \{0\}$ and the total number of non-zero numbers is larger than 0. This means that $\xi_{i_k}, \varepsilon_{j_{k,\phi}} \in \mathbb{R}^+$ are random positive numbers independent with regard to attribute values and that the sign of $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,\phi}}$ is determined by the sign of $(b_{j_{k,\phi}} - a_{i_k})$. Therefore, $CIPE_0$ is correct.

D. Security Proof

Theorem 1. *$CIPE_0$ is secure with leakage function $\mathcal{L}$ in the known ciphertext model.*

Proof. We first consider the the security of our scheme in the single-attribute setting, where a data record D_i or a query $\mathcal{Q}_j$ is associated with only one attribute. We adopt a simulation-based proof similar to the one used in [13], [17]. Let Adv be an adversary, and let Sim be a simulator that can simulate an output $V' = (\mathcal{I}', \mathbb{T}')$ with leakage function $\mathcal{L}$.

• Sim randomly picks two invertible matrices $\mathbf{M}_1', \mathbf{M}_2' \in \mathbb{R}^{d \times d}$, a bit string S' of length d, and a permutation σ' on $[d]$. Sim sets $SK' = (sk', \sigma')$ where $sk' = (\mathbf{M}_1', \mathbf{M}_2', S')$.

• To generate $\mathcal{I}'$, Sim generates a pair of d-dimensional vectors $\mathbf{p}_i'$ for $i \in [n]$ as follows: 1) It constructs a d-dimensional vector $\mathbf{p}_i$ where each element is initialized to 0. 2) It chooses two random numbers $c_i \in \mathbb{R}^+$ and $a_i \in \mathbb{R}$ and sets $\mathbf{p}_i[1] = c_i a_i$, $\mathbf{p}_i[2] = c_i$, $\mathbf{p}_i[3] = 1$, and $\mathbf{p}_i[4] = a_i$. While generating random attribute values, the order of $a_1, \ldots, a_n$ should be consistent with that of attribute values in $\mathcal{D}$. 3) It shuffles elements in $\mathbf{p}_i$ with σ'. 4) It runs $KNN.EncI(\mathbf{p}_i, sk')$ to output $\mathbf{p}_i'$. Therefore, $\mathcal{I}' = \{\mathbf{p}_1', \ldots, \mathbf{p}_n'\}$.

• To generate $\mathbb{T}'$, Sim constructs two pairs of d-dimensional vectors $(\mathbf{q}_{j_l}', \mathbf{q}_{j_u}')$ for $j \in [q]$ as follows: 1) It constructs two d-dimensional vectors $\mathbf{q}_{j_l}$ and $\mathbf{q}_{j_u}$, where each element is initialized to 0. 2) It constructs empty sets R. For $i \in [n]$, if $D_i \bowtie \mathcal{Q}_j$, it puts a_i in set R. 3) It chooses two random numbers $\bar{c}_{j_l} \in \mathbb{R}^+$ and $b_{j_l} \in \mathbb{R}$ such that $a^- < b_{j_l} \le \min \mathrm{R}$, where $\min \mathrm{R}$ is the minimal attribute value in R and a^- is the first attribute value in $\{a_1, \ldots, a_n\}$ smaller than $\min \mathrm{R}$. It sets $\mathbf{q}_{j_l}[1] = -1$, $\mathbf{q}_{j_l}[2] = b_{j_l}$, $\mathbf{q}_{j_l}[3] = \bar{c}_{j_l} b_{j_l}$, and $\mathbf{q}_{j_l}[4] = -\bar{c}_{j_l}$. 4) It chooses two random numbers $\bar{c}_{j_u} \in \mathbb{R}^+$ and $b_{j_u} \in \mathbb{R}$ such that $\max \mathrm{R} \le b_{j_u} < a^+$, where $\max \mathrm{R}$ is the maximal attribute value in R and a^+ is the first attribute value in $\{a_1, \ldots, a_n\}$ larger than $\max \mathrm{R}$. It sets $\mathbf{q}_{j_u}[1] = -1$, $\mathbf{q}_{j_u}[2] = b_{j_u}$, $\mathbf{q}_{j_u}[3] = \bar{c}_{j_u} b_{j_u}$, and $\mathbf{q}_{j_u}[4] = -\bar{c}_{j_u}$. 5) It shuffles elements in $\mathbf{q}_{j_l}$ and $\mathbf{q}_{j_u}$ with σ'. 6) It runs $KNN.EncQ(\mathbf{q}_{j_l}, sk')$ and $KNN.EncQ(\mathbf{q}_{j_u}, sk')$ to output $\mathbf{q}_{j_l}'$ and $\mathbf{q}_{j_u}'$, respectively. Therefore, $\mathbb{T}' = \{(\mathbf{q}_{1_l}', \mathbf{q}_{1_u}'), \ldots, (\mathbf{q}_{q_l}', \mathbf{q}_{q_u}')\}$.

The indistinguishability of indexes and trapdoors is based on the indistinguishability of KNN and the introduced randomness. As shown in the correctness analysis, the outputs of the $Search_0$ algorithm look like random values that reveal only the order information between data attribute values and query attribute values. Therefore, we claim that no PPT adversary can distinguish V' from V. Our basic construction $CIPE_0$ is built based on the single-attribute setting, and it is secure with leakage function $\mathcal{L}$ in the known ciphertext model. ■

E. Discussion

How to hide distance and equality information. In our construction, $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,\phi}} = (\xi_{i_k} + \varepsilon_{j_{k,\phi}})(b_{j_{k,\phi}} - a_{i_k})$, where $k \in [m]$ and $\phi \in \{l, u\}$. Since $\xi_{i,k}, \varepsilon_{j_{k,\phi}} \in \mathbb{R}^+$ may be any positive numbers, the results of the inner products have no direct relation to the distance between $b_{j_{k,\phi}}$ and a_{i_k}. For example, in Fig. 1, D_2 with x-coordinate value 350 has the minimal distance between 330. However, the result of $\mathbf{p}_{2_1} \cdot \mathbf{q}_{1,l}$ in Fig. 2 is -98, the absolute value of which is the second largest. Therefore, random positive numbers are helpful to conceal distance information. Furthermore, given two data records D_i and $D_{i'}$ with the same attribute value on A_k, i.e., $a_{i_k} = a_{i'_k}$, we have $\mathbf{p}_{i_k} \cdot \mathbf{q}_{j_{k,\phi}} \ne \mathbf{p}_{i'_k} \cdot \mathbf{q}_{j_{k,\phi}}$, as long as $\xi_{i_k} \ne \xi_{i'_k}$ where $k \in [m]$ and $\phi \in \{l, u\}$. For example, the attribute values of A_1 for D_4, D_5 in Fig. 1 are the same (i.e., both x-coordiate values are 450). However, in Fig. 2, $\mathbf{p}_{4_1} \cdot \mathbf{q}_{1,l} = -72$ and $\mathbf{p}_{5_1} \cdot \mathbf{q}_{1,l} = -144$; their inner products generate different results. Therefore, the equality of attribute values will not be leaked due to random positive numbers.

Resistant to ciphertext only attacks (COA). The work that is most similar to our basic CIPE construction can be found in [24]. In their scheme, a data value a_i and a query value b_j are encoded as vectors $\mathbf{p}_i$ and $\mathbf{q}_j$, respectively, such that $\mathbf{p}_i \cdot \mathbf{q}_j = \xi_i \varepsilon_j (b_j - a_i)$ where $\xi_i, \varepsilon_j \in \mathbb{R}^+$ are randomness introduced into $\mathbf{p}_i$ and $\mathbf{q}_j$ respectively. However, as proven in [25], their scheme is vulnerable to COA:

$$\frac{\mathbf{p}_i \cdot \mathbf{q}_j}{\mathbf{p}_i \cdot \mathbf{q}_{j'}} = \frac{\xi_i \varepsilon_j (b_j - a_i)}{\xi_i \varepsilon_{j'} (b_{j'} - a_i)} = \frac{\varepsilon_j}{\varepsilon_{j'}} \times \frac{(b_j - a_i)}{(b_{j'} - a_i)} \qquad (1)$$

Algorithm 3 Query Matrix Encoding

```
Require: $s \in [2 \sim d]$, $r, t \in \mathbb{R}$ for $r + t > 0$, query vector $\mathbf{q}_{j_{k,\phi}} \in \mathbb{R}^d$ for $k \in [m]$ and $\phi \in \{l, u\}$
Ensure: query matrix $\mathbf{Q}_{j_{k,\phi}} \in \mathbb{R}^{d \times s}$
1: construct a matrix $\mathbf{Q}_{j_{k,\phi}} \in \mathbb{R}^{d \times s}$ and initialize each element of $\mathbf{Q}_{j_{k,\phi}}$ with 0
2: for $\iota \in [d]$ do
3:     set $r\mathbf{q}_{j_{k,\phi}}[\iota]$ at a random position of the $\iota$-th row of $\mathbf{Q}_{j_{k,\phi}}$ with the constraint of Condition 1
4:     fill the remaining elements in the $\iota$-th row of $\mathbf{Q}_{j_{k,\phi}}$ with random numbers with the constraint of Condition 2
```

Given $b_j > b_{j'}$, function $\frac{\varepsilon_j}{\varepsilon_{j'}} \times \frac{(b_j - x)}{(b_{j'} - x)}$ is strickly increasing, and thus the order of an arbitrary pair of data values can be obtained by comparing the result of Eq. 1. In $CIPE_0$, $\mathbf{p}_i \cdot \mathbf{q}_j = (\xi_i + \varepsilon_j)(b_j - a_i)$, where neither ξ_i nor ε_j can be eliminated by calculating Eq. 1 or by other operations. Therefore, the adversary cannot obtain an univariate function to initiate COA.

How to achieve vectors of constant length. The length of vectors in $CIPE_0$ grows linearly with the number of attributes m. To create constant-length vectors, our main idea is to divide a d-dimensional vector into m sub-vectors, where the k-th sub-vector of length $\frac{d}{m}$ is built for attribute A_k. For $k \in [m]$, we run Alg. 1 (resp. Alg. 2) for the k-th sub-vector, then we encrypt it with algorithm $KNN.EncI$ (resp. $KNN.EncQ$). However, d must be large enough so that each sub-vector contains sufficient randomness. Furthermore, a larger m introduces more *false positives* to the search results, since the combined effect of m sub-vectors may conceal the mismatching of a single one. Therefore, there is a tradeoff using this improvement.

How to achieve design goals. 1) *High efficiency.* The $Trapdoor_0$ algorithm generates trapdoors purely based on current queries and the $Search_0$ algorithm is independently run by the cloud server. Therefore, our scheme is stateless and non-interactive. 2) *High scalability.* The most expensive operation in the $Search_0$ algorithm is determining the vector inner product, which costs only 1.4 microseconds to calculate the inner product of two 128-dimensional vectors. While conducting experiments on a massive data set, we build an R-tree like that in [22] to achieve a faster-than-linear search time. 3) *Enhanced security.* The main reason that OPE schemes are vulnerable to inference attacks is that their ciphertexts reveal either order or frequency information about the underlying plaintexts immediately. In our scheme, ciphertexts are random vectors leaking nothing about corresponding plaintexts, and plaintext order is not exposed until a range query is performed. Therefore, it can effectively resist inference attacks (Our last design goal, *enhanced privacy*, is achieved in Section V).

V. Advanced CIPE (CIPE$_S$)

A. Main Idea

From the construction of KNN [20], we know that the inner product of an index vector and a query vector can be calculated from their encrypted forms: $\mathbf{p} \cdot \mathbf{q} = \mathbf{p}'_{|\alpha|} \cdot \mathbf{q}'_{|\alpha|} + \mathbf{p}'_{|\beta|} \cdot \mathbf{q}'_{|\beta|}$, where only d variables are unknown to the cloud server, i.e., d elements of the index vector $\mathbf{p}$. As proven in Yao et al. [23], an adversary with d query vectors, and their encryptions can recover index vectors by solving linear equations.

To resist chosen plaintext attacks, we extend query vectors to random query matrices, thereby adding noises to the results of inner product. The hardness for initiating such attacks comes from the negligible probability of constructing d correct equations. For all index vectors we have:

$$\mathbf{p} \cdot \mathbf{q} \neq \mathbf{p'}_{|\alpha|} \cdot \mathbf{q}'_{|\alpha|} + \mathbf{p'}_{|\beta|} \cdot \mathbf{q}'_{|\beta|}. \qquad (2)$$

Let $\mathbf{M}[i][j]$ denote the element at the i-th row and the j-th column of matrix $\mathbf{M}$, and let $\mathbf{M}[i][*]$ ($\mathbf{M}[*][j]$) denote a vector of elements in the i-th row (j-th column) of $\mathbf{M}$. Our solution is based on the following key techniques:

Encoding query matrices. Given two random numbers $r, t \in \mathbb{R}$ such that $r + t > 0$, a query vector $\mathbf{q}_{j_{k,\phi}}$ generated by Alg. 2 is encoded as a $d \times s$ matrix $\mathbf{Q}_{j_{k,\phi}}$ by Alg. 3. For $\iota \in [d]$, Alg. 3 sets $\mathbf{q}_{j_{k,\phi}}[\iota]$ at a random position of the ι-th row of $\mathbf{Q}_{j_{k,\phi}}$ and fills other positions with random numbers while the following conditions are satisfied:

Condition 1. *Each column of matrix $\mathbf{Q}_{j_{k,\phi}}$ contains at least one element of vector $\mathbf{q}_{j_{k,\phi}}$.*

Condition 2. *The sum of the random numbers at the ι-th row, denoted as δ_ι, is equal to $t\mathbf{q}_{j_{k,\phi}}[\iota]$.*

The way to determine matching. Given a d-dimensional index vector $\mathbf{p}_{i_k}$ output by Alg. 1 and two $d \times s$ query matrices $\mathbf{Q}_{j_{k,l}}$ and $\mathbf{Q}_{j_{k,u}}$ output by Alg. 3, two intermediate vector $\mathbf{R}_{k,l}, \mathbf{R}_{k,u} \in \mathbb{R}^d$ is obtained by calculating $\mathbf{p}_{i_k}^T \star \mathbf{Q}_{j_{k,l}}$ and $\mathbf{p}_{i_k}^T \star \mathbf{Q}_{j_{k,u}}$, respectively. Given the sum of elements in $\mathbf{R}_{k,l}$ and $\mathbf{R}_{k,u}$, denoted as $\gamma_{k,l}$ and $\gamma_{k,u}$, respectively, we have $D_i \bowtie \mathcal{Q}_j$ if $\gamma_{k,l} \leq 0$ and $\gamma_{k,u} \geq 0$ for $k \in [m]$.

B. Advanced Construction

The main differences between $CIPE_0$ and $CIPE_S$ lie in the algorithms $Trapdoor_S$ and $Search_S$ as follows:

• $Trapdoor_S(\mathcal{Q}_j, SK) \rightarrow T_{\mathcal{Q}_j}$: She chooses a random integer $s \in [2 \sim d]$ and two random numbers $r, t \in \mathbb{R}$ such that $r + t > 0$. Given query $Q_{j_k} = [b_{j_{k,l}}, b_{j_{k,u}}]$ where $k \in [m]$, the data owner generates query vectors $\mathbf{q}_{j_{k,l}}, \mathbf{q}_{j_{k,u}} \in \mathbb{R}^d$ by running Alg. 2. Then, she constructs query matrices $\mathbf{Q}_{j_{k,l}}, \mathbf{Q}_{j_{k,u}} \in \mathbb{R}^{d \times s}$ based on $\mathbf{q}_{j_{k,l}}, \mathbf{q}_{j_{k,u}}$ by running Alg. 3. For $\iota \in [s]$, she runs algorithm $KNN.EncQ$ to encrypt the ι-th column of matrix $\mathbf{Q}_{j_{k,l}}$ and $\mathbf{Q}_{j_{k,u}}$, and outputs $\mathbf{Q}'_{j_{k,l}} = (\mathbf{Q}'_{|\alpha|j_{k,l}}, \mathbf{Q}'_{|\beta|j_{k,l}})$ and $\mathbf{Q}'_{j_{k,u}} = (\mathbf{Q}'_{|\alpha|j_{k,u}}, \mathbf{Q}'_{|\beta|j_{k,u}})$, respectively. Finally, she sets $T_{\mathcal{Q}_j} = (T_{Q_{j_1}}, \ldots, T_{Q_{j_m}})$ where $T_{Q_{j_k}} = (\mathbf{Q}'_{j_{k,l}}, \mathbf{Q}'_{j_{k,u}})$ for $k \in [m]$.

• $Search_S(\mathcal{I}_i, T_{\mathcal{Q}_j}) \rightarrow \{0, 1\}$: For $k \in [m]$, the cloud server computes $\mathbf{R}_{k,l} = \mathbf{p'}^T_{|\alpha|i_k} \star \mathbf{Q}'_{|\alpha|j_{k,l}} + \mathbf{p'}^T_{|\beta|i_k} \star \mathbf{Q}'_{|\beta|j_{k,l}} = \mathbf{p}^T_{i_k} \star \mathbf{Q}_{j_{k,l}}$ and $\mathbf{R}_{k,u} = \mathbf{p'}^T_{|\alpha|i_k} \star \mathbf{Q}'_{|\alpha|j_{k,u}} + \mathbf{p'}^T_{|\beta|i_k} \star \mathbf{Q}'_{|\beta|j_{k,u}} = \mathbf{p}^T_{i_k} \star \mathbf{Q}_{j_{k,u}}$ It calculates the sum of each vector by setting $\gamma_{k,l} = \sum_{\iota=1}^{s} \mathbf{R}_{k,l}[\iota]$ and $\gamma_{k,u} = \sum_{\iota=1}^{s} \mathbf{R}_{k,u}[\iota]$. It outputs 1 if $\gamma_{k,l} \leq 0$ and $\gamma_{k,u} \geq 0$ for $k \in [m]$; otherwise, it outputs 0.

C. Correctness Analysis

The correctness of $CIPE_S$ is based on the assumption that the introduced randomness in a query matrix will not impact the positive and negative property of results. Given an index vector $\mathbf{p}$ and a query vector $\mathbf{q}$, a query matrix $\mathbf{Q}$ of $\mathbf{q}$ is output by Alg. 3. Let $x_{i,j}$ denote the element in the i-th row and j-th column of matrix $\mathbf{Q}$ for $i \in [d], j \in [s]$. The intermediate vector in the $Search_S$ algorithm can be derived as follows:

$$\begin{bmatrix} \boxed{-0.15} & 1.105 & -1 \\ -2 & \boxed{0} & 2 \\ -2 & -32.65 & \boxed{49.5} \\ \boxed{0} & -3 & 3 \\ \boxed{99} & -63.3 & -6 \\ \boxed{99} & -300 & 230.7 \\ \boxed{-0.3} & 0.21 & 0 \\ \boxed{-0.3} & 0 & 0.21 \end{bmatrix}$$

(a) $\mathbf{Q}_{1,l}$ (d=8, s=3).

$$\begin{bmatrix} \boxed{0} & 1 & 0 & -1 & 2 & 4 & -3 & -3 \\ -2 & \boxed{-0.02} & 0.006 & -1 & 1 & -4 & 5 & 1 \\ 3.1 & 1 & \boxed{0} & -2.58 & 1.9 & -2 & -3 & -1 \\ -5 & -3 & -2 & \boxed{8.6} & 0 & 10 & 0 & 0 \\ 2 & 4 & -3 & -258 & \boxed{860} & -3 & -1 & 1 \\ 2.5 & 3.5 & -4 & -1 & -258 & \boxed{860} & 1.5 & -2.5 \\ 3 & -6 & 3 & -0.5 & -1.5 & 0.6 & \boxed{-2} & 2 \\ 3 & -3 & 0 & 1 & -4 & 3 & 0.6 & \boxed{-2} \end{bmatrix}$$

(b) $\mathbf{Q}_{1,u}$ (d=8, s=8).

$$\boxed{\begin{array}{l} \sum(\mathbf{p}^T{}_{1_1} \star \mathbf{Q}_{1,l}) = 1.62, \sum(\mathbf{p}^T{}_{1_1} \star \mathbf{Q}_{1,u}) = 20.2 \\ \sum(\mathbf{p}^T{}_{2_1} \star \mathbf{Q}_{1,l}) = -8.82, \sum(\mathbf{p}^T{}_{2_1} \star \mathbf{Q}_{1,u}) = 493.92 \\ \sum(\mathbf{p}^T{}_{3_1} \star \mathbf{Q}_{1,l}) = -3.78, \sum(\mathbf{p}^T{}_{3_1} \star \mathbf{Q}_{1,u}) = 4.62 \\ \sum(\mathbf{p}^T{}_{4_1} \star \mathbf{Q}_{1,l}) = -6.48, \sum(\mathbf{p}^T{}_{4_1} \star \mathbf{Q}_{1,u}) = -3.08 \\ \sum(\mathbf{p}^T{}_{5_1} \star \mathbf{Q}_{1,l}) = -12.96, \sum(\mathbf{p}^T{}_{5_1} \star \mathbf{Q}_{1,u}) = -19.88 \end{array}}$$

(c) Results of the sum of matrix multiplication.

Fig. 3. Working process of $CIPE_S$. In sample matrices $\mathbf{Q}_{1,l}$ ($\mathbf{Q}_{1,u}$), the boxes filled with blue color contain elements in query vector $r\mathbf{q}_{1,l}$ ($r\mathbf{q}_{1,u}$).

$$\begin{aligned} \mathbf{R} &= \mathbf{p}^T \star \mathbf{Q} = (\mathbf{p} \cdot \mathbf{Q}[*][1], \cdots, \mathbf{p} \cdot \mathbf{Q}[*][s]) \\ &= (\sum_{i=1}^{d} \mathbf{p}[i]x_{i,1}, \sum_{i=1}^{d} \mathbf{p}[i]x_{i,2}, \cdots, \sum_{i=1}^{d} \mathbf{p}[i]x_{i,s}) \end{aligned}$$

The flag value γ can be calculated:

$$\begin{aligned} \gamma &= \sum_{i=1}^{d} \mathbf{p}[i]x_{i,1} + \cdots + \sum_{i=1}^{d} \mathbf{p}[i]x_{i,s} \\ &= \mathbf{p}[1](x_{1,1} + \cdots + x_{1,s}) + \cdots + \mathbf{p}[d](x_{d,1} + \cdots + x_{d,s}) \end{aligned} \tag{3}$$

Based on **Condition** 1-2, Eq. 3 can be converted to Eq. 4:

$$\begin{aligned} \gamma &= \mathbf{p}[1](r\mathbf{q}[1] + \delta_1) + \cdots + \mathbf{p}[d](r\mathbf{q}[d] + \delta_d) \\ &= \mathbf{p}[1](r\mathbf{q}[1] + t\mathbf{q}[1]) + \cdots + \mathbf{p}[d](r\mathbf{q}[d] + t\mathbf{q}[d]) \\ &= (r + t)(\mathbf{p}[1]\mathbf{q}[1] + \cdots + \mathbf{p}[d]\mathbf{q}[d]) = (r + t)\mathbf{p} \cdot \mathbf{q} \end{aligned} \tag{4}$$

Since $(r + t) > 0$, the sign of the flag value γ is the same as that of $\mathbf{p} \cdot \mathbf{q}$, and our advanced construction is correct.

Example 2. Given query vectors $\mathbf{q}_{1,l}$ and $\mathbf{q}_{1,u}$ as shown in Fig. 2-(b), sample query matrices $\mathbf{Q}_{1,l}$ and $\mathbf{Q}_{1,u}$ are shown in Fig. 3-(a) and Fig. 3-(b), respectively. For $\mathbf{q}_{1,l}$, we set $r = 0.3$ and $t = -0.21$ and have $\sum(\mathbf{p}_{i_1}^T \star \mathbf{Q}_{1,l}) = 0.09(\mathbf{p}_{i_1} \cdot \mathbf{q}_{1,l})$ for $i \in [5]$. For $\mathbf{q}_{1,u}$, we set $r = 2$ and $t = -0.6$ and have $\sum(\mathbf{p}_{i_1}^T \star \mathbf{Q}_{1,u}) = 1.4(\mathbf{p}_{i_1} \cdot \mathbf{q}_{1,u})$ for $i \in [5]$. Therefore, the results in Fig. 3-(c) are consistent with those in Fig. 2-(c).

D. Security Proof

Theorem 2. *$CIPE_S$ is secure with leakage function $\mathcal{L}$ in the known background model.*

Proof (sketch). The purpose of extending a query vector to a random matrix is to add random noises to the search results so that Eq. 2 holds for all index vectors. First, we show the security of the flag value γ as follows: from Eq. 4, we know that $\gamma = (r + t)\mathbf{p} \cdot \mathbf{q}$, where $(r + t)$ is a random positive number having nothing to do with vectors. Both $(r + t)$ and $\mathbf{p} \cdot \mathbf{q}$ may be non-integer values. Therefore, it is impossible for the adversary Adv to decompose $\mathbf{p} \cdot \mathbf{q}$ from γ.

In $CIPE_S$, query matrices output by the $Trapdoor_S$ algorithm are encrypted with KNN, and algorithms $KeyGen_S$ and $SecureInx_S$ are constructed in the same way as the algorithms in the basic construction. Therefore, the privacy objectives we define in Section II-B are preserved in $CIPE_S$. Our main security concern is that the $Search_S$ algorithm leaks an intermediate vector $\mathbf{R}$, which may be used to infer certain sensitive information about underlying data sets and submitted queries. We show *the security of the intermediate result* $\mathbf{R}$ as follows: for $\iota \in [s]$, the ι-th element of $\mathbf{R}$ is the result of the inner product of $\mathbf{p}$ and $\mathbf{Q}[*][\iota]$. In our construction, each column of a matrix contains $(d - s + 1)$ query values at most and $(s - 1)$ random numbers at least. Although the sum of the random numbers at each row is related to the query value ($\delta_\iota = t\mathbf{q}[\iota]$), the random numbers in each column are independent from both the index and query vectors. Therefore, Adv cannot infer any useful information from $\mathbf{R}$ directly. Now, let us consider Adv constructing linear equations from an arbitrary combination of $\mathbf{R}[1], \ldots, \mathbf{R}[s]$. If Adv adds up $\mathbf{R}[i], \ldots, \mathbf{R}[j]$ where $i, j \in [s]$ and $|i - j| < s - 1$, the number of unknown variables is larger than the number of linear equations. If Adv adds up $\mathbf{R}[1], \ldots, \mathbf{R}[s]$, it obtains the flag value γ, whose security we have already proved. Therefore, our advanced CIPE construction is secure. ■

E. Discussion

Resistant to known plaintext attacks (KPA). As proven in [25], the scheme in [24] is also vulnerable to KPA. Given two query values b_0, b_1, the adversary Adv obtains $\hat{a}_i a_i = a_i\alpha - \beta + \hat{a}_i b_1$ from Eq. 1, where $\hat{a}_i = \frac{\mathbf{p}_i \cdot \mathbf{q}_0}{\mathbf{p}_i \cdot \mathbf{q}_1} = \frac{\varepsilon_0}{\varepsilon_1} \times \frac{b_0 - a_i}{b_1 - a_i}$, $\alpha = \frac{\varepsilon_0}{\varepsilon_1}$, and $\beta = \alpha b_0$. Given three data values a_0, a_1, a_2, Adv can obtain α, β by solving linear equation:

$$(\alpha, \beta, b_1) \begin{pmatrix} a_0 \; a_1 \; a_2 \\ -1 - 1 - 1 \\ \hat{a}_0 \; \hat{a}_1 \; \hat{a}_2 \end{pmatrix} = (\hat{a}_0 a_0, \hat{a}_1 a_1, \hat{a}_2 a_2) \tag{5}$$

Then, Adv recovers plaintexts a_i by calculating $a_i = \frac{b_1 \hat{a}_i - b_0 \alpha}{\hat{a}_i - \alpha}$. In $CIPE_0$, neither ξ_i nor ε_j can be cancelled by calculating Eq. 1 or by other operations, and the number of unknown variables grows as the number of linear equations grows. Therefore, Adv cannot calculate Eq. 5 to initiate KPA.

The impact of parameter s. The number of columns s in query matrix mainly impact the performance of algorithms $Trapdoor_S$ and $Search_S$. As shown in Fig. 4, the larger the s, the higher the cost. In terms of security level, even if $s = 2$, the adversary Adv cannot obtain any useful information from a single column of $\mathbf{R}$ directly. We believe that even for this low value of s, there is a sufficient measure of security provided.

VI. Evaluation

In this section, we evaluate the performance of our CIPE scheme in terms of computational and communication costs. To show the effectiveness of our scheme, we compare it with the ideally secure OPE scheme proposed in [6] (denoted by mOPE), which requires $O(\log n)$ rounds of communication.

A. Parameter Setting

Experiments are conducted on a local machine running the Microsoft Windows 7 Ultimate operating system with an Intel Core i5 CPU running at 3.3GHz and 32GB memory. The programs are implemented in Java compiled using Eclipse 4.3.2. In terms of mOPE, AES-ECB-128 is employed for data encryption with 192-bit ciphertexts (i.e., 128 bits for an AES ciphertext and 64 bits for a mutable encoding).

To validate the feasibility of our CIPE scheme in practice, we conduct a performance evaluation on Gowalla data set

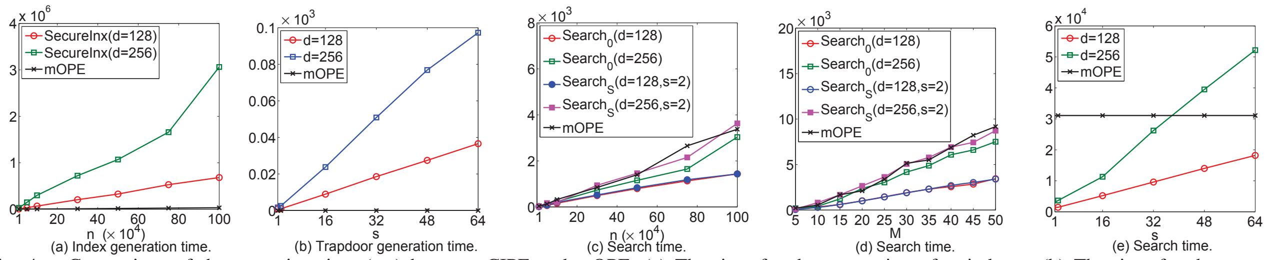


Fig. 4. Comparison of the execution time (ms) between CIPE and mOPE. (a) The time for the encryption of n indexes. (b) The time for the generation of a trapdoor for matrix of s columns. (c) The time for searching n files with fixed $M = 25$. (d) The searching time for rectangles of size M with fixed $n = 1,000,000$. (e) The searching time for query matrices of s columns with fixed $n = 1,000,000$ and $M = 25$.

from the SNAP project[1]; this includes a total of 6,442,890 location check-ins collected from 196,591 users. We extract $n = [10,000 \sim 1,000,000]$ distinct data from Gowalla data set, where each data is described by $m = 2$ attributes, i.e., the (latitude, longitude) pair. For security considerations, the size of the vectors is set to $d = \{128, 256\}$ for both $CIPE_0$ and $CIPE_S$. In $CIPE_S$, the number of matrix columns s is set to $[2 \sim 64]$. We evaluate the search efficiency with rectangles of different sizes (the edge length M is set to $[5 \sim 50]$). At least 20 random 2-dimensional range queries are generated and evaluated on each size to minimize deviation.

B. Experiment Results

Computational costs. To analyze computational complexity, we only consider the most expensive operations related to matrices in our constructions. In terms of key generation, both algorithms, $KeyGen_0$ and $KeyGen_S$, generate two $d \times d$ invertible matrices, the complexity of which is $O(d^2)$. In our experiments, the costs of both algorithms under the settings $d = 128$ and $d = 256$ are about 31ms and 47ms, respectively.

To generate secure indexes, the costs of $SecureInx_0$ and $SecureInx_S$, collectively called $SecureInx$ in Fig. 4-(a), are the same. For $j \in [m]$, $SecureInx$ runs the $KNN.EncI$ algorithm to generate a pair of d-dimensional vectors, the complexity of which is $O(d^2)$. Therefore, the cost of building n indexes for a m-dimensional data set is $O(nmd^2)$. The experiment results are consistent with our analysis results. For example, when d increases from 128 to 256, the execution time increases from 682s to 3,062s for encrypting $n = 1,000,000$ indexes; when $d = 128$ and n ranges from 10,000 to 1,000,000, the execution time costs from 6s to 682s. Compared with mOPE scheme, the performance of our $SecureInx$ algorithm is slightly worse. For example, the execution time to encrypt an attribute value is about 0.04ms in mOPE, and about 0.2ms and 0.4ms in our scheme when $d = 128$ and $d = 256$, respectively. However, the data encryption is a one-time cost.

The $Trapdoor_0$ algorithm runs the $KNN.EncQ$ algorithm to generate m pairs of d-dimensional encrypted vectors, the complexity of which is $O(md^2)$. For $Trapdoor_S$, we encrypt each column separately, and thus its complexity is $O(smd^2)$. As shown in Fig. 4-(b), the execution time for generating a trapdoor is very fast. For example, when $s = 1$, the execution time is equivalent to that of $Trapdoor_0$ and is about 0.8ms and 1.7ms when $d = 128$ and $d = 256$, respectively.

The $Search_0$ algorithm runs the $KNN.Cal$ algorithm to calculate vector inner products with a complexity of $O(md)$. The $Search_S$ algorithm runs the $KNN.Cal$ algorithm for each column, and its cost is $O(mds)$. If we evaluate a trapdoor on the data set sequentially, the cost to find all matched data records for the algorithms $Search_0$ and $Search_S$ is $O(nmd)$ and $O(nmds)$, respectively. To accelerate the query speed, we build a 4-ary R-tree from n data records, where the leaf node points to a data record, and each non-leaf node represents a rectangle. Inspired by the work in [22], mutable encoding is used to label paths in the R-tree. Given a search rectangle B we test whether rectangles in non-leaf nodes intersect with B and test whether data records in the leaf nodes fall within B. Therefore, the search cost is also impacted by the size of search rectangle M. The experiment results are shown in Figs. 4(c)-(e). These figures show that our $Search$ algorithm (which calcultes inner products) is more efficient than mOPE (which performs AES decryption) in most cases; they also show that a faster-than-linear search time is achieved and that either a larger M or a larger s incurs a larger execution time.

Communication costs. In both algorithms, $SecureInx_0$ and $SecureInx_S$, an attribute value is encoded to a pair of d-dimensional vectors. Therefore, the cost of transmitting n indexes for a m-dimensional data set is $O(nmd)$. In the search phase, the $Trapdoor_0$ algorithm encodes an attribute value to a pair of d-dimensional vectors and the $Trapdoor_S$ algorithm encodes an attribute value to a pairs of $d \times s$ matrices. Their costs are $O(md)$ and $O(msd)$, respectively. The cost of the transmission of trapdoors is trivial (e.g., about 12KB in $Trapdoor_0$ when $d = 256$). Therefore, Table II only lists the comparison results of index sizes (compressed sizes) before and after building a R tree. Since each element in a vector/matrix is 8 bytes, our scheme incurs more communication costs than mOPE that generates a ciphertext of 128-bit length. However, this is a one-time cost that can be further reduced using existing compression techniques (e.g. 7zip can save communication costs by 30%). With the ever increasing availability of the bandwidth, it takes less than 10 minutes to complete the transmission over a 100Mbps network.

VII. Related Work

Existing research most related to ours can be found in OPE and ORE. OPE guarantees that $Enc(x) > Enc(y)$ if $x > y$, allowing for performing range queries directly on the ciphertexts. Agrawal et al. [21] first introduce OPE, which relies on heuristics, but lacks a formal security analysis. Subsequently, Boldyreva et al. [4] define the formal notion of IND-OCPA as *the ideal security* for OPE, in which the ciphertexts reveal no additional information beyond the order of plaintexts. Since then, a lot of work has been conducted to construct ideally secure OPE schemes. For example, Papa et al. [6] construct

[1]http://snap.stanford.edu/data/loc-gowalla.html

TABLE II. COMPARISON OF INDEX SIZES

$n(\times 10^4)$	Linear search			R-tree		
	$d = 128$	$d = 256$	mOPE	$d = 128$	$d = 256$	mOPE
1	91.2MB (32.2MB)	190MB (67.6MB)	673KB (269.2KB)	138.8MB (49MB)	290MB (102.4MB)	1.003MB (410.736KB)
5	464MB (188MB)	974MB (391MB)	3.29MB (1.316MB)	703.36MB (289MB)	1.45GB (595MB)	5.05MB (2.02MB)
10	935MB (380MB)	1.89GB (781MB)	6.58MB (2.632MB)	1.38GB (580MB)	2.9GB (1.157GB)	10.02MB (4.008MB)
30	2.71GB (1.084GB)	5.71GB (2.284GB)	19.7MB (7.88MB)	4.13GB (1.652GB)	8.69GB (3.476GB)	30.2MB (12.08MB)
50	4.55GB (1.82GB)	9.52GB (3.808GB)	32.9MB (13.16MB)	6.85GB (2.74GB)	14.38GB (5.752GB)	50.1MB (20.04MB)
75	6.72GB (2.688GB)	14.2GB (5.68GB)	49.3MB (19.72MB)	10.08GB (4.032GB)	21.5GB (8.6GB)	74.7MB (29.88MB)
100	9.07GB (3.628GB)	18.8GB (7.52GB)	65.8MB (26.32MB)	13.63GB (5.452GB)	28.5GB (11.4GB)	99.2MB (39.68MB)

TABLE III. COMPARISON WITH PREVIOUS WORK

	Efficiency	Scalability	Security	Privacy
Ref. [6]–[8]		✓		✓
Ref. [9]		✓	✓	✓
Ref. [14]–[16]	✓		✓	✓
Ref. [13], [17]	✓	✓		✓
Our CIPE scheme	✓	✓	✓	✓

a mutable OPE scheme which requires $O(\log n)$ interactions between clients and servers where n is the number of data records. As a trade-off, Kerschbaum et al. [7] reduce the communication costs to $O(1)$, but require $O(n)$ client storage. More recently, Kerschbaum et al. [8] introduce a notion of frequency-hiding OPE (FHOPE) by introducing randomness to ciphertexts. Roche et al. [9] improve the security and efficiency of FHOPE by applying lazy indexing and partial ordering techniques. However, existing ideally secure OPE schemes are inefficient (either stateful or interactive). Besides efficiency, as shown in [10]–[12], almost all existing OPE schemes are vulnerable to inference attacks even if they are ideally secure.

The notion of ORE, in which a function takes two ciphertexts as inputs and outputs the ordering of the underlying plantexts, is first introduced by Boneh et al. [14]. However, all existing constructions of ORE that achieve ideal security rely on very strong cryptographic primitives [14]–[16], such as multilinear maps and obfuscate functions. Therefore, these approaches are far from practically viable. Very recently, Chenette et al. [17] provide an efficient ORE construction based on pseudorandom functions. However, their scheme resorts to weaker security which leaks information beyond order, i.e., the index of the first bit position that differs between two plaintexts. Lewi et al. [13] construct an ORE scheme with improved security by leaking only the first block that differs between two plaintexts. However, the ciphertext length in their scheme grows linearly with the plaintext domain. In summary, the aforementioned schemes only partially address our design goals. The comparison between our work and previous OPE and ORE schemes is shown in Table III.

VIII. CONCLUSION

In this paper, we study the problem of multi-attribute range queries on encrypted data, and we propose a CIPE scheme to achieve secure and effective search services in cloud computing. Experiment results demonstrate that our scheme is extremely efficient for secure range queries on large-scale data sets. As part of our future work, we will try to implement our scheme directly in database management systems.

ACKNOWLEDGMENT

This work was supported in part by NSFC grants 61632009; and NSF grants CNS 1757533, CNS1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

REFERENCES

[1] M. Xiao, J. Wu, S. Zhang, and Y. Yu, "Secret-sharing-based secure user recruitment protocol for mobile crowdsensing," *Proc. of INFOCOM*, 2017.

[2] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, J. Cao, and D. Zhang, "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM Transactions on Networking*, 2017.

[3] K. Xie, X. Li, X. Wang, J. Cao, G. Xie, J. Wen, D. Zhang, Z. Qin, "On-line anomaly detection with high accuracy," *IEEE/ACM Transactions on Networking*, 2018.

[4] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-preserving symmetric encryption," in *Proc. of EUROCRYPT*, 2009.

[5] A. Boldyreva, N. Chenette, and A. O'neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Proc. of CRYPTO*, 2011.

[6] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. of S&P*, 2013.

[7] F. Kerschbaum and A. Schröpfer, "Optimal average-complexity ideal-security order-preserving encryption," in *Proc. of CCS*, 2014.

[8] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proc. of CCS*, 2015.

[9] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, "POPE: Partial order preserving encoding," in *Proc. of CCS*, 2016.

[10] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. of CCS*, 2015.

[11] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?," in *Proc. of CCS*, 2016.

[12] P. Grubbs, K. Sekniqi, V. Bindschaedler, et al., "Leakage-abuse attacks against order-revealing encryption," in *Proc. of S&P*, 2017

[13] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. of CCS*, 2016.

[14] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, et al.,"Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *Proc. of EUROCRYPT*, 2015.

[15] P. Ananth and A. Jain, "Indistinguishability obfuscation from compact functional encryption," in *Proc. of CRYPTO*, 2015.

[16] S. Kim, K. Lewi, A. Mandal, and D. J. Wu, "Function-hiding inner product encryption is practical," *IACR Cryptology ePrint Archive*, 2016.

[17] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," in *Proc. of FSE*, 2016.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. of Eurocrypt*, 1999.

[19] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of STOC*, 2009.

[20] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. of SIGMOD*, 2009.

[21] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. of SIGMOD*, 2004.

[22] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *Proc. of INFOCOM*, 2016.

[23] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. of ICDE*, 2013.

[24] P. Karras, A. Nikitin, M. Saad, R. Bhatt, D. Antyukhov, S. Idreos, "Adaptive indexing over encrypted numeric data," in *Proc. of SIGMOD*, 2016.

[25] C. Horst, R. Kikuchi, and K. Xagawa, "Cryptanalysis of comparable encryption in SIGMOD'16," in *Proc. of SIGMOD*, 2017.