Secure Phrase Search for Intelligent Processing of Encrypted Data in Cloud-Based IoT

Meng Shen, *Member, IEEE*, Baoli Ma, Liehuang Zhu, *Member, IEEE*, Xiaojiang Du, *Senior Member, IEEE*, and Ke Xu, *Senior Member, IEEE*

***Abstract*—Phrase search allows retrieval of documents containing an exact phrase, which plays an important role in many machine learning applications for cloud-based Internet of Things (IoT), such as intelligent medical data analytics. In order to protect sensitive information from being leaked by service providers, documents (e.g., clinic records) are usually encrypted by data owners before being outsourced to the cloud. This, however, makes the search operation an extremely challenging task. Existing searchable encryption schemes for multikeyword search operations fail to perform phrase search, as they are unable to determine the location relationship of multiple keywords in a queried phrase over encrypted data on the cloud server side. In this paper, we propose P3, an efficient privacy-preserving phrase search scheme for intelligent encrypted data processing in cloud-based IoT. Our scheme exploits the homomorphic encryption and bilinear map to determine the location relationship of multiple queried keywords over encrypted data. It also utilizes a probabilistic trapdoor generation algorithm to protect users' search patterns. Thorough security analysis demonstrates the security guarantees achieved by P3. We implement a prototype and conduct extensive experiments on real-world datasets. The evaluation results show that compared with existing multikeyword search schemes, P3 can greatly improve the search accuracy with moderate overheads.**

***Index Terms*—Artificial intelligence, cloud, encrypted data, Internet of Things (IoT), phrase search.**

I. Introduction

PHRASE search, which allows users to search for sentences or documents containing a specific phrase that consists of a set of consecutive keywords [1], serves as an important building block in many machine learning applications for cloud-based Internet of Things (IoT) [27]. For instance, it can be applied to intelligent clinical data analytics collected from medical IoT devices, which retrieves medical records related to a certain disease (e.g., myocardial infarction) and feeds machine learning algorithms to obtain portent symptoms of the disease. It can also be applied to the emerging entity-oriented search [21], which identifies the records within which the exact description of an *entity* (e.g., person or event) occurs. The resulting records can be utilized for situation assessment and intelligent decision making. Another application scenario refers to the semantic search in knowledge graphs, which searches for entities with semantic similarity (e.g., titles, positions, and interests) and provides input signals to machine learning models for recommendation of products, news, and advertisements.

The combination of cloud computing and IoT enables powerful processing of data beyond individual IoT devices with limited capabilities. This, however, raises a great concern about the security and privacy of IoT data stored in the cloud, as untrusted cloud service providers may get access to sensitive data or even result in data leakage accidents [25], [26]. In order to protect data privacy, data owners can opt to encrypt their sensitive data before outsourcing the storage of the data to remote cloud servers. For instance, a healthcare company may store their encrypted patients' records in the cloud, and allow only the authorized users to perform phrase search over these records. This naturally imposes a requirement on the cloud-based search engine to perform phrase search operations over encrypted data.

Many schemes [2], [4], [5], [7], [8], [11], [14]–[16], [18]–[20], [23], [29]–[35], [38] have been proposed to enable efficient search operations over encrypted textual data, as summarized in Table I. Existing solutions to the single-keyword and multikeyword search problems cannot be used to perform phrase search over encrypted documents, because they are unable to determine the positional[1] relationship of the keywords composing a phrase in the encrypted environment. For instance, the conjunctive keyword search scheme [4] will return a document if it contains each keyword at least once, regardless of whether these keywords appear consecutively as a phrase. Therefore, if we use this scheme for phrase search, we would end with inaccurate results (see Section VI).

Manuscript received May 1, 2018; revised August 3, 2018; accepted September 12, 2018. Date of publication September 20, 2018; date of current version May 8, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0803405, in part by the National Natural Science Foundation of China under Grant 61602039, Grant 61472212, and Grant 61872041, in part by the EU Marie Curie Actions CROWN under Grant FP7-PEOPLE-2013-IRSES-610524, and in part by the CCF-Tencent Open Fund WeBank Special Funding. *(Corresponding author: Liehuang Zhu.)*

M. Shen, B. Ma, and L. Zhu are with the Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: shenmeng@bit.edu.cn; baolimasmile@bit.edu.cn; liehuangz@bit.edu.cn).

X. Du is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: dxj@ieee.org).

K. Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xuke@mail.tsinghua.edu.cn).

Digital Object Identifier 10.1109/JIOT.2018.2871607

[1]We use the terminologies of *positional information* and *location information* interchangeably in this paper.

TABLE I
SUMMARY OF PRIOR SOLUTIONS AND P3

Solutions	Multiple Keywords	Phrase Search	Single Interaction	T.T.P. Free
Single keyword search [7, 18, 33]	×	×	√	√
Multi-keyword search [4, 23, 29, 31] [5, 11, 19, 32, 34]	√	×	√	√
Phrase search [20, 30]	√	√	×	√
Phrase search [38]	√	√	√	×
P3	√	√	√	√

There are a limited number of studies targeting the phrase search problem over encrypted data [20], [30], [38]. These solutions, however, generally involve notable limitations as shown in Table I, e.g., by either requiring resource-consuming multiple rounds of client-server interactions, or relying on a trusted third-party (TTP) for search result refinement on the behalf of the client.

Since the client-side IoT devices usually have constrained computing and storage resources, we aim at developing a phrase search scheme that achieves all of the attributes listed in Table I. The main challenge is to enable cloud servers to make a judgement on whether the keywords occurring in an encrypted document are consecutive or not, without leaking sensitive information.

In this paper, we propose P3, a new privacy-preserving phrase search scheme over cloud-based encrypted data. We take advantage of the inverted index structure to build a secure index that achieves greater flexibility and efficiency. The inverted index is one of the most popular and efficient index structures for plaintext search. Compared with the diverse self-designed index structures [4], [5], [23], [29], [32], the inverted index structure can improve retrieval efficiency and scalability in practice. To tackle the challenge of determining the positional relationship of queried keywords over encrypted data, we resort to the homomorphic encryption and bilinear map, which enables the client to obtain exact search results from a single interaction with the cloud server. As the phrase search is a special case of multikeyword search, our solution can also perform conjunctive multikeyword search efficiently.

The main contributions of this paper are as follows.

1) We propose a secure single-interaction phrase search scheme that enables phrase search over encrypted data in cloud-based IoT, without relying on a TTP.
2) We employ the combination of homomorphic encryption and bilinear map to determine the pairwise positional relationship of queried keywords on the cloud server side. It can be used as a building block in other relevant application scenarios.
3) We implement a prototype of P3 and conduct extensive experimental evaluation using real-world datasets. Results demonstrate that P3 greatly improves the search accuracy with moderate overheads.

The rest of this paper is organized as follows. We summarize the related work in Section II and present the problem formulation in Section III. We describe the proposed scheme in Section IV and provide the security analysis in Section V. We evaluate P3 through extensive experiments in Section VI and discuss the limitations in Section VII. Finally, we conclude this paper in Section VIII.

II. RELATED WORK

The privacy-preserving data processing problem has attracted great research attention during the last decade [12], [17], [36], [37]. The secure searchable encryption problem was first addressed by Song *et al.* [28], which was index-free and could merely support exact single keyword search. In order to extend the functionality and efficiency of searchable encryption, follow-ups have proposed various schemes that support single keyword search [7], [18], [33] and exact or fuzzy multikeyword search [4], [5], [11], [15], [16], [19], [23], [29], [31], [32], [34], by using either self-designed indexes or the typical inverted index structure. Several attempts have been taken to extend the fuzzy multikeyword search scheme to support phrase search, either by treating a predefined phrase (e.g., network security) as a single keyword [6] or introducing a TTP server on the client side [38].

Tang *et al.* [30] proposed a phrase search construction over encrypted cloud data, but failed to implement and evaluate their proposal in real-world application scenarios. For each individual phrase recognition, this construction needed two rounds of communications between the client and the server, and also required a large number of trapdoors generated by the client. Poon and Miri [20] proposed a phrase search scheme with relatively low storage and computational overhead. However, they failed to present a complete threat model, a security definition, or a reasonable security proof. Therefore, it remains unclear about the privacy guarantees provided by the proposed method.

In contrast to the existing phrase search solutions, the phrase search scheme proposed in this paper is a single-interaction scheme without a TTP. Therefore, it can achieve higher flexibility and lower communication overhead.

III. PROBLEM FORMULATION

In this section, we formally define the secure phrase search problem in intelligent processing of encrypted data. We denote several keywords whose locations in the documents are consecutive are a *phrase*. We denote a keyword collection of the documents and their corresponding document identifier and location information as an *index*, and an encrypted index as a *secure index*. We refer to a searched phrase as a *query* and an encrypted query as a *trapdoor*.

A. System Model

The privacy-preserving phrase search system over encrypted data involves three entities, namely an *IoT data owner*, a *cloud server*, and one or multiple *users*, as illustrated in Fig. 1. The data owner generates a secure searchable index for the document set and outsources the secure index along with the encrypted document set to the cloud server. When an authorized user, say Alice, performs a phrase search over

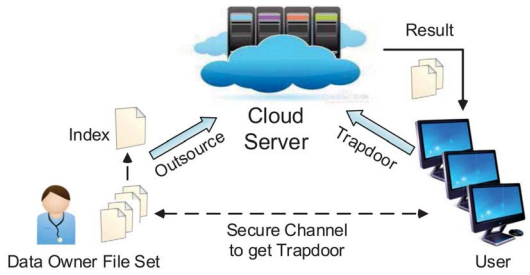

Fig. 1. System model of cloud-based phrase search over encrypted data.

the encrypted documents, she first acquires the corresponding trapdoor from the data owner through the search control mechanism (e.g., broadcast encryption [7]), and then submits the trapdoor to the cloud server. Upon receiving Alice's trapdoor, the cloud server executes the predesigned search algorithms and replies to the user with the corresponding set of encrypted documents as the search results. Finally, the user decrypts the received documents with the help of the data owner.

We assume that both the user and the data owner have limited computation and storage capacities on a practical basis. Existing key management mechanisms [9], [10], [22] can be employed to manage the encryption capabilities of authorized users.

The above scheme is formally defined as follows.

Definition 1 (Privacy-Preserving Phrase Search Scheme): A privacy-preserving phrase search scheme consists of the following polynomial time algorithms.

1) *KeyGen(τ, d):* Let τ and d be security parameters as inputs of KeyGen($\cdot$), and a master key Mk be an output.
2) *IndexGen(Mk, Γ):* It executes on the data owner side and takes the master key Mk and the document collection Γ as inputs and the secure index $\widehat{\mathbb{I}}$ as an output.
3) *TrapdoorGen(Mk, Q):* Given the master key Mk and a query Q from a user, it outputs the secure trapdoor $\mathbb{T}_Q$. This process is also performed on the data owner side.
4) *Query($\widehat{\mathbb{I}}$, $\mathbb{T}_Q$):* Given the secure index $\widehat{\mathbb{I}}$ and the trapdoor $\mathbb{T}_Q$, it performs search operations on the cloud server side and returns query results.

B. Security Model

Similar to the existing searchable encryption solutions [31], [32], we consider the cloud server as an *honest-but-curious* adversary. That is, the cloud server would honestly follow the predesigned phrase search protocols and correctly provide the corresponding services to users, but, it may be curious about the contents of the documents and attempt to learn additional information by analyzing the trapdoor and indexes. For instance, it would infer the keywords in the index and trapdoors, as well as their locations in the documents.

Motivated by [4], [13], [23], and [32], we consider the following two threat models with different attack capabilities, depending on the sensitive information that can be obtained by the cloud server.

1) *Known Ciphertext Model:* The cloud server can only access the encrypted document set and the corresponding secure index that are outsourced by the data owner, and the trapdoors submitted by users. The cloud server is also capable of recording the search history, such as the search results in terms of encrypted documents.
2) *Known Background Model:* In this stronger model, the cloud server is assumed to be aware of more facts than what can be known in the *known ciphertext model*. In particular, the cloud server can learn the statistical information, such as keyword frequency in the document set. Furthermore, given such statistical information, the cloud server may infer the keywords in a queried phrase.

Our scheme aims at protecting privacy associated with the phrase search operation, which consists of three types of privacy, namely the document set privacy, the index privacy, and the trapdoor privacy. The document set privacy can be easily achieved by encrypting the documents using a block cipher, such as AES, before outsourcing them to the cloud server. Therefore, in this paper we focus on the latter two aspects, which are described as follows.

1) *Index Privacy:* Since the secure index can be regarded as a representation of the encrypted documents, any further information (e.g., keywords) should not be deduced from the index by the cloud server, except for the relationship between a trapdoor and its corresponding search results. In general, index privacy refers to the information of keywords, document identifiers, and keyword locations. Here, the keyword location privacy is guaranteed once the location information of all keywords is protected. We assume that the relationship between the keyword locations can be revealed to the cloud server, which does not go against the keyword location privacy.
2) *Trapdoor Unlinkability:* The trapdoors are used by the cloud server to perform matches with the secure index. Intuitively, the trapdoors should not reveal any valuable information (e.g., search frequency). The unlinkability means that the cloud server is unable to associate a trapdoor with the corresponding search phrase, i.e., the trapdoors generated for the same plaintext phrase should be different in multiple queries (e.g., queries submitted by multiple users or at different time periods).

C. Definition and Notation

Now, we introduce the main notations and the rest of the notations are summarized in Table II.

1) Γ: A finite set of documents stored in plaintext, denoted as $\Gamma = (f_1, f_2, \ldots, f_m)$, where f_i is the ith document.
2) W: A finite set of keywords extracted from the document set Γ, denoted as $W = (w_1, w_2, \ldots, w_\mu)$, where w_i is the ith keyword in W.
3) $\mathbb{I}$: An inverted index of the document set Γ, denoted as $\mathbb{I} = (\mathbb{I}_{w_1}, \mathbb{I}_{w_2}, \ldots, \mathbb{I}_{w_\mu})$, where $\mathbb{I}_{w_i}$ is the inverted list corresponding to w_i. For each inverted list, we have $\mathbb{I}_{w_i} = (w_i, \Omega_{i1}, \Omega_{i2}, \ldots, \Omega_{ik})$, where Ω_{ij} represents the jth entity in $\mathbb{I}_{w_i}$. Let $\Omega_{ij} = (f_{ij}, \Lambda_{ij})$ be a tuple of the document identifier $f_{ij} \in \Gamma$ $(1 \leq j \leq k)$ and the location identifier Λ_{ij}. Λ_{ij} is a list of keyword locations in f_{ij}, which is denoted by $\Lambda_{ij} = \langle l_{j1}, l_{j2}, \ldots, l_{jt} \rangle$. Here, l_{jr} $(1 \leq r \leq t)$ is the location where the keyword w_i appears in the document f_{ij}.

TABLE II
NOTATIONS FOR PHRASE SEARCH SCHEME

Notation	Definition
$\widehat{\mathbb{I}}$	The encrypted form of $\mathbb{I}$
Q	A query consisting of multiple keywords in W
$\|Q\|$	Number of keywords in the query Q
$\mathbb{T}_Q$	Trapdoor of the query Q
$\|\mathbb{I}\|$	Number of items in the inverted index
$\|\mathbb{I}_{w_i}\|$	Number of documents containing the keyword w_i
η	Maximum number of entries in a secure inverted list

D. Preliminaries

Bilinear map is a function combining elements of two groups (e.g., $\mathbb{G}_1$ and $\mathbb{G}_2$) to yield an element of a third group (e.g., $\mathbb{G}_T$). We now briefly review it. For simplicity, we consider a special case where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of a finite order n, and g be a generator of $\mathbb{G}$. A bilinear map e is a function in

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T \tag{1}$$

with a useful property: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$, yet $e(g, g)$ is a generator of $\mathbb{G}_T$.

Homomorphic encryption is a cryptography primitive that allows us to perform operations over encrypted data without knowing the secret key or decrypting the data. Boneh *et al.* [3] proposed a homomorphic encryption scheme based on finite groups of composite order that supported a bilinear map, which can be briefly described in the following three steps.

1) *Key Generation:* Assume that $\mathbb{G}$ and $\mathbb{G}_T$ are two (multiplication) cyclic groups of finite order n, and e is a bilinear map. Let g and u be two random generators of $\mathbb{G}$, and p, q be two big primes satisfying $n = pq$. Set $h = u^q$, then let $pk = (n, \mathbb{G}, \mathbb{G}_T, e, g, h)$ and $sk = p$.
2) *Encryption:* A message m can be encrypted to its ciphertext c as follows:

$$c = g^m h^r \in \mathbb{G}$$

where r is randomly picked in $\{0, 1, \ldots, n-1\}$.
3) *Decryption:* A ciphertext c is decrypted as follows:

$$c^p = \left(g^m h^r\right)^p = g^{mp} u^{rpq} = \left(g^p\right)^m \pmod{n}.$$

Let $\hat{g} = g^p$. One needs to compute the discrete log of c^p base $\hat{g}$ to recover m.

This scheme has the additive homomorphism over the encrypted data feature. Given the ciphertext $E(a)$ and $E(b)$, we can get the result of $a + b$ by $E(a) \cdot E(b)$, i.e., $E(a + b) = E(a) \cdot E(b)$. This feature allows us to calculate the sum of two numbers by their ciphertexts without decryption.

IV. SECURE PHRASE SEARCH FOR INTELLIGENT PROCESSING OF ENCRYPTED DATA

This section presents the proposed privacy-preserving phrase search scheme over encrypted data.

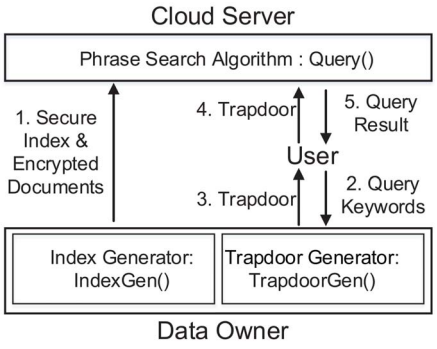

Fig. 2. Structure and workflow of the proposed scheme P3.

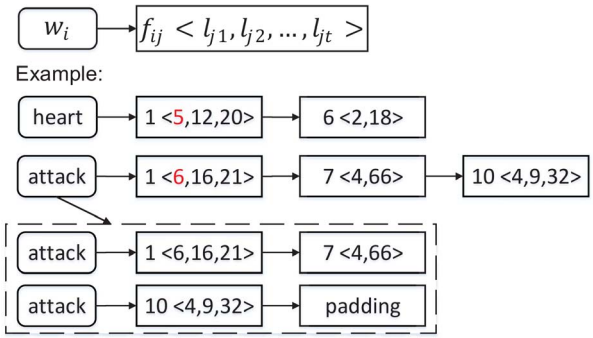

Fig. 3. Example of the inverted index (encryptions are not shown).

A. System Overview

The structure and workflow of the proposed scheme, P3, are depicted in Fig. 2, which mainly consists of the following three modules.

1) *Index generator*, which is executed on the data owner side. It takes the documents as the input and outputs the corresponding secure index, as well as the encrypted documents.
2) *Trapdoor generator*, which is also executed on the data owner side. Given a user's queried phrase, it generates the corresponding secure trapdoor and replies to the user.
3) *Phrase search algorithm*, which is executed on the cloud server side. Upon receiving a trapdoor from a user, it performs a phrase search procedure over the secure index and returns the search results.

In order to support phrase search, we leverage the inverted index structure and store the keyword locations along with the document identifier, as shown in Fig. 3 (see Section III-C for explanations of notations). In the example illustrated in Fig. 3, there are two files containing the keyword *heart*, namely Files 1 and 6. More precisely, the locations of *heart* in File 1 are 5, 12, and 20, respectively.

The phrase search procedure can be described as follows. When the cloud server receives the trapdoor for a specific phrase query from a user, it first locates the inverted lists for the queried keywords, and then finds the documents that contain all of the queried keywords. After that, the cloud server identifies whether the locations of the keywords are consecutive and returns only the relevant documents that contain the exact phrase. As shown in Fig. 3, File 1 should be returned if the user queries the phrase "heart attack."

Algorithm 1 EncKeywordForIndex(·)

Input: $\{w_i, K, S, M_1, M_2\}$, where K is secret key of PRF π and S, M_1, and M_2 are the secret keys of the secure kNN technique. Define $S(i)$ as the i-th bit in S.

Output: The encrypted keyword identifier $\widetilde{Z}_{w_i}$ in the index.

1: Construct a vector $\widetilde{B} = \{\pi(K, w_i)^0, \ldots, \pi(K, w_i)^{d-1}\}^T$, where d is the length of S and $\pi(\cdot)$ is a secure PRF primitive.
2: **for** $i \leftarrow 1$ **to** d **do**
3: **if** $S(i) = 1$ **then**
4: Split $\widetilde{B}(i)$ randomly into $\widetilde{B}^a(i)$ and $\widetilde{B}^b(i)$ with $\widetilde{B}^a(i) + \widetilde{B}^b(i) = \widetilde{B}(i)$.
5: **else**
6: Set both $\widetilde{B}^a(i)$ and $\widetilde{B}^b(i)$ to $\widetilde{B}(i)$.
7: **end if**
8: **end for**
9: Encrypt $\widetilde{B}$ as $M_1^T\widetilde{B}^a$ and $M_2^T\widetilde{B}^b$.
10: Set $\widetilde{Z}_{w_i} = \{M_1^T\widetilde{B}^a, M_2^T\widetilde{B}^b\}$.
11: **return** $\widetilde{Z}_{w_i}$

It is easy to perform phrase search over plaintexts. However, it is difficult for the server to determine whether or not the keywords occur in documents as a phrase, given the encrypted location information of each pair of keywords. To tackle this challenge, we propose a series of designs based on the homomorphic encryption [3] and bilinear groups. We also utilize the widely used secure kNN method [4], [23], [32] to achieve trapdoor unlinkability.

B. Building Blocks

As described in Section IV-A, we should ensure privacy in the index generation, the trapdoor generation, and the phrase search procedures. We now introduce basic building blocks to achieve these goals.

1) Keyword Representation in the Secure Index and the Trapdoor: We utilize a similar technique as in [23] to achieve the goals of index privacy and trapdoor unlinkability.

Our design is based on the following observation. Given a polynomial function $f(x)$ of degree m, which is denoted by $f(x) = (x-t_1)(x-t_2)\cdots(x-t_m) = a_0+a_1x+\cdots+a_mx^m$, we can extract the coefficients to form a vector $A = \{a_0, a_1, \ldots, a_m\}$. We can also construct another vector $B = \{t^0, t^1, \ldots, t^m\}^T$, where $t \in \{t_1, t_2, \ldots, t_m\}$. Note that t^m represents t to the power of m. Since t is a root of $f(x)$, we have $A^T \cdot B = 0$.

Based on the above knowledge, for any single keyword we construct two vectors, A and B, as its representations in the trapdoor and the index, respectively. Then, we can know if a keyword in the trapdoor matches a keyword in the index by checking whether $A^T \cdot B = 0$. Hence, we now focus on constructing these two vectors for private-preserving matching.

To generate the encrypted keyword identifier $\widetilde{Z}_{w_i}$ for each keyword $w_i \in W$, we utilize the secure kNN technique, as depicted in Algorithm 1. The algorithm includes two steps, where the first step is to create the vector $\widetilde{B}$ (line 1), and the second step is to obtain the encrypted keyword identifier $\widetilde{Z}_{w_i}$ by splitting $\widetilde{B}$ randomly into two vectors $\widetilde{B}^a(i)$ and $\widetilde{B}^b(i)$ (lines 2–9). According to the value of each element in S, $\widetilde{B}^a(i)$ and $\widetilde{B}^b(i)$ are assigned with different values. We refer the readers to [4], [23], and [32] for the rationale of secure kNN.

Algorithm 2 EncKeywordForTrapdoor(·)

Input: $\{w_i, K, S, M_1, M_2\}$, where K is secret key of PRF π and S, M_1, and M_2 are the secret keys of the secure kNN technique. Define $S(i)$ as the i-th bit in S.

Output: The encrypted keyword identifier $\widetilde{Y}_{w_i}$ in the trapdoor.

1: Construct a keyword vector $\Phi = \{w_i, w'_1, \ldots, w'_{d-2}\}$, where d is the length of S and $\{w'_1, \ldots, w'_{d-2}\}$ are $d-2$ dummy keywords.
2: Get a vector $\widetilde{\Phi} = \{\pi(K, w_i), \pi(K, w'_1), \ldots, \pi(K, w'_{d-2})\}$, where d is the length of S and $\pi(\cdot)$ is a secure PRF primitive.
3: Construct a polynomial function of degree $d-1$ as $f(x) = (x - \pi(K, w_i)) \times (x - \pi(K, w'_1)) \times \cdots \times (x - \pi(K, w'_{d-2})) = a_0 + a_1x + \cdots + a_{d-1}x^{d-1}$.
4: Extract the coefficients of $f(x)$ to form the query vector $\widetilde{A} = \{a_0, a_1, \ldots, a_{d-1}\}^T$.
5: **for** $i \leftarrow 1$ **to** d **do**
6: **if** $S(i) = 0$ **then**
7: Split $\widetilde{A}(i)$ randomly into $\widetilde{A}^a(i)$ and $\widetilde{A}^b(i)$, where $\widetilde{A}^a(i) + \widetilde{A}^b(i) = \widetilde{A}(i)$.
8: **else**
9: Set both $\widetilde{A}^a(i)$ and $\widetilde{A}^b(i)$ to $\widetilde{A}(i)$.
10: **end if**
11: **end for**
12: Encrypt $\widetilde{A}$ as $M_1^{-1}\widetilde{A}^a$ and $M_2^{-1}\widetilde{A}^b$.
13: Set $\widetilde{Y}_{w_i} = \{M_1^{-1}\widetilde{A}^a, M_2^{-1}\widetilde{A}^b\}$.
14: **return** $\widetilde{Y}_{w_i}$

To construct a secure trapdoor for a query Q, we also utilize the secure kNN technique to construct the encrypted keyword identifier $\widetilde{Y}_{w_i}$ for each keyword $w_i \in Q$, as described in Algorithm 2. It consists of two steps, where the first step (lines 1–4) is to create the vector $\widetilde{A}$, and the second step (lines 5–12) is to spilt $\widetilde{A}$ to obtain the encrypted keyword identifier $\widetilde{Y}_{w_i}$.

Based on the above constructions, given an encrypted keyword identifier $\widetilde{Y}_{w_i}$ in a trapdoor, the cloud server can locate an inverted list with an encrypted keyword identifier $\widetilde{Z}_{w_i}$, by checking whether $\widetilde{Y}_{w_i}^T \cdot \widetilde{Z}_{w_i} = 0$.

The correctness of this construction is illustrated by

$$\begin{aligned}\widetilde{Y}_{w_i}^T \cdot \widetilde{Z}_{w_i} &= \left\{M_1^{-1}\widetilde{A}^a, M_2^{-1}\widetilde{A}^b\right\}^T \cdot \left\{M_1^T\widetilde{B}^a, M_2^T\widetilde{B}^b\right\} \\ &= \left(\widetilde{A}^a\right)^T\left(M_1^{-1}\right)^T M_1^T\widetilde{B}^a + \left(\widetilde{A}^b\right)^T\left(M_2^{-1}\right)^T M_2^T\widetilde{B}^b \\ &= \left(\widetilde{A}^a\right)^T\widetilde{B}^a + \left(\widetilde{A}^b\right)^T\widetilde{B}^b \\ &= \widetilde{A}^T \cdot \widetilde{B}. \end{aligned} \tag{2}$$

The secure kNN method is vulnerable to linear analysis, and this means that the cloud server may launch the linear analysis on a large number of pairs of keyword identifiers between the secure index and the trapdoors. To address this limitation, we adopt dummy keywords in the procedure of trapdoor generation (lines 1–3 in Algorithm 2). Therefore, for the same keyword over multiple queries, we can obtain a different coefficient vector $\widetilde{A}$ (line 4 in Algorithm 2). Furthermore, due to the property of the secure KNN technique, we can perform various splittings over a coefficient vector $\widetilde{A}$. Hence, our construction is secure against linear analysis.

2) Phrase Recognition: To protect the keyword location privacy, we encrypt the keyword location through the homomorphic encryption scheme introduced in Section III-D.

Note that in our scheme, we only publish $(n, \mathbb{G}, \mathbb{G}_T, e)$ to the cloud server as the public key. Assume that a and b represent locations of two different keywords in a same document. Without loss of generality, we also assume that $a < b$. If these two keywords are consecutive, we have $a - b + 1 = 0$, i.e., $b - a = 1$. To determine the relationship between a and b on the basis of their ciphertexts $g^a h^{r_1}$ and $g^b h^{r_2}$, the cloud server sets $x = a - b + 1$ and transforms this problem to an equivalent problem of determining whether x is the ciphertext of 0, as

$$\begin{aligned} E(x) &= E(a - b + 1) \\ &= g^a h^{r_1} \cdot \left(g^b h^{r_2}\right)^{-1} \cdot g^1 h^{r_3} \\ &= g^x h^r \end{aligned} \tag{3}$$

where $g^1 h^{r_3}$ represents the ciphertext of 1.

Then, the cloud server further determines the relationship between a and b depending on the result of

$$e\left(E(x), \lambda^p\right) = e\left(g^x h^r, \lambda^p\right) \tag{4}$$

where $\lambda \in \mathbb{G}$, p is the private key, and λ^p is the dispersal factor that cannot be an identity of $\mathbb{G}$.

Until now, the cloud server has known $g^x h^r$ and λ^p. To eliminate the random value r, it then computes $e(g^x h^r, \lambda^p)$ by bilinear maps. Note that a and b represent consecutive locations if and only if the result of (4) is equal to 1, as $e(g^x h^r, \lambda^p) = e(g^0 h^r, \lambda^p) = e(h^r, \lambda^p) = e(h, \lambda)^{rp} = e(h^{rp}, \lambda) = e(1, \lambda) = 1$.

The idea of such a design comes from the fact that we can eliminate the existence of the random value r for $(h^r)^p = u^{rpq} = u^{rn} = 1 \pmod n$. However, since the phrase recognition procedure is performed by the cloud server, a user cannot send p to the cloud server directly. Therefore, the user randomly picks an element $\lambda \in \mathbb{G}$ and sends λ^p to the cloud server. Since λ and p are both secret, the cloud server cannot infer p from λ^p.

Now, we briefly discuss the construction of the phrase recognition process. First, at a high level, we want to protect the keyword location information, rather than the keyword location relationship in the phrase search. This is because revealing the keyword location relationship is inevitable to perform phrase recognition. Second, the recognition method can determine an arbitrary interval for two integers. In other words, if we want to know whether the interval between two locations a and b is d, we can just send $g^d h^r$ to the cloud server, where r is a random number. In addition, the ciphertexts for the same d over multiple queries are different. This property can prevent the cloud server from inferring the interval d, because the cloud server cannot know the real value of d even if it learns that a and b satisfy a certain relationship.

Note that this application scenario is different from the well-known secure multiparty computation (i.e., SMC). In the setting of SMC, set of parties with private inputs wish to compute a function of their inputs while revealing nothing but the result of the function, which is used for many practical applications such as exchange markets. SMC is a collaborative computing problem that solves the privacy preserving problem among a group of mutually untrusted participants. Thus, the SMC schemes are fully secure, they protect the location relationship between keywords against the cloud server. As a result, the phrase recognition procedure can only be performed on either the user side or the data owner side, which sacrifices the main benefit of offloading computation to cloud servers. Therefore, we make a compromise that revealing the relationship between keyword locations for better efficiency.

3) Division and Padding of Inverted List: To protect the keyword privacy, it is necessary to hide its appearance frequency in each document. We divide each inverted list to make it contain η documents. Then, if the length of an (original or divided) inverted list is smaller than η, we perform a padding for the remaining entries. In the example shown in Fig. 3, we choose $\eta = 2$ and divide "attack" into two inverted lists, where the second list has a padding entry. More precisely, each entity that we pad consists of an invalid document identifier and some random numbers as fake keyword locations.

In order to distinguish these invalid document identifiers from the valid ones, we use a counter that is initialized as -1 and gradually decrease it by 1 for each padded document identifier. Due to the encryption of the invalid and valid document identifiers, the cloud server cannot tell which document identifer is invalid.

Since we utilize the probabilistic encryption, a same keyword w_i will have different ciphertexts (i.e., the encrypted keyword identifier $\widetilde{Z}_{w_i}$). Therefore, from the perspective of the cloud server, it seems that each inverted list corresponds to a unique keyword. In the performance evaluation, we select η as the frequency median of all the keywords in the document set. We leave the exploration of optimal η to the future work.

C. Scheme Details

This section describes the privacy-preserving phrase search scheme in detail, which consists of four components.

1) KeyGen(τ, d): Given the security parameters τ and d, the data owner generates the master key and the public key by taking the following steps.

1) Generate two random τ-bit big primes p and q, and set $n = p * q$. Construct the bilinear groups $\mathbb{G}$ and $\mathbb{G}_T$ and the bilinear map e using the method introduced in [3]. Then, pick two random generators, g and u, from $\mathbb{G}$, and set $h = u^q$. Note that h is a random generator of the subgroup of $\mathbb{G}$ of order p.
2) Randomly generate a d-bit binary string S and two $d \times d$ invertible matrices M_1 and M_2. Let $S(i)$ be the ith bit of S.
3) Let π be a secure pseudorandom function (PRF) primitive and generate a τ-bit secret key K.
4) Let ν be a secure pseudorandom permutation (PRP) primitive and generate a τ-bit secret key U.

The data owner keeps the tuple $(p, g, h, K, U, S, M_1, M_2)$ as the master key (i.e., Mk) and the tuple $(n, \mathbb{G}, \mathbb{G}_T, e)$ as the public key (i.e., pk), which is published to the cloud server.

2) IndexGen(Mk, Γ): The data owner builds the secure inverted index in the following steps.

1) Extract a distinct keyword collection W of size μ from the document collection Γ. For each keyword $w_i \in W (1 \le i \le \mu)$, build the inverted list $\mathbb{I}_{w_i}$ as described in

Fig. 3, which consists of the identifiers of documents that contain keyword w_i along with all the keyword locations, i.e., $\mathbb{I}_{w_i} = (w_i, \Omega_{i1}, \Omega_{i2}, \ldots, \Omega_{ik})$, where $\Omega_{ij} = (f_{ij}, \Lambda_{ij})$ and $\Lambda_{ij} = \langle l_{j1}, l_{j2}, \ldots, l_{jt} \rangle$, $1 \le j \le k$, $1 \le r \le t$. Set the inverted index $\mathbb{I} = \{\mathbb{I}_{w_1}, \mathbb{I}_{w_2}, \ldots, \mathbb{I}_{w_\mu}\}$.

2) For each $\mathbb{I}_{w_i} \in \mathbb{I}$, encrypt the document identifier by $\nu(U, f_{ij})$ and encrypt the keyword locations.
More precisely, for each location $l_{jy} \in \Lambda_{ij}$, pick a random number $r_{jy} \in \{0, 1, \ldots, n-1\}$; then, we have the encrypted location c_{jy} as described by

$$c_{jy} = g^{l_{jy}} h^{r_{jy}}. \tag{5}$$

To hide keyword frequencies, we should guarantee that different keywords have the same frequency. Hence, the data owner should further process $\mathbb{I}_{w_i}$ via division and padding. If $\lfloor |\mathbb{I}_{w_i}| \ mod \ \eta \rceil = 0$, divide $\mathbb{I}_{w_i}$ into $|\mathbb{I}_{w_i}|/\eta$ individual inverted lists, which are defined as $\{\mathbb{I}_{w_{i1}}, \mathbb{I}_{w_{i2}}, \ldots, \mathbb{I}_{w_{it}}\}$, $1 \le t \le |\mathbb{I}_{w_i}|/\eta$. While if $\lfloor |\mathbb{I}_{w_i}| \ mod \ \eta \rceil \ne 0$, divide $\mathbb{I}_{w_i}$ into $1 + |\mathbb{I}_{w_i}|/\eta$ individual inverted lists, which are defined as $\{\mathbb{I}_{w_{i1}}, \mathbb{I}_{w_{i2}}, \ldots, \mathbb{I}_{w_{it}}\}$, $1 \le t \le 1 + |\mathbb{I}_{w_i}|/\eta$. For $\mathbb{I}_{w_{it}}$ where $t = 1 + |\mathbb{I}_{w_i}|/\eta$, the data owner pads some random dummy document identifiers and binary strings of length $|c_{jy}|$ to make sure that the keyword document frequency is η.

3) For each inverted list $\mathbb{I}_{w_{it}}$ of the keyword w_{it}, encrypt the keyword w_{it} to obtain the encrypted keyword identifier $\widetilde{Z}_{w_{it}}$ using Algorithm 1. Then, update w_{it} with $\widetilde{Z}_{w_{it}}$. We now get the secure inverted index $\widehat{\mathbb{I}} = \{\{\widehat{\mathbb{I}}_{w_{it}}\}\}$, where $1 \le i \le \mu$.

3) TrapdoorGen(Mk, Q): Given a query Q, a user can retrieve the corresponding trapdoor from the data owner, which takes the following steps.

1) For each $w_j \in Q$, $1 \le j \le |Q|$, generate the encrypted query keyword identifier $\widetilde{Y}_{w_j}$ using Algorithm 2.
2) We assume that the search distance β is 1. Pick a random number $r \in [0, n-1]$, and then compute the ciphertext of 1

$$\mathbb{C} = g^1 h^r. \tag{6}$$

3) Randomly pick an element $\lambda \in \mathbb{G}$, and then compute the dispersal factor, $\psi = \lambda^p$, where λ^p is not an identity of $\mathbb{G}$.

The trapdoor $\mathbb{T}_Q = \{\{\widetilde{Y}_{w_j}\}, \mathbb{C}, \psi\}$, where $1 \le j \le |Q|$.

4) Query($\widehat{\mathbb{I}}, \mathbb{T}_Q$): Once the cloud server receives the trapdoor from the user, the cloud server first locates the inverted lists corresponding to the queried keywords, by checking whether $\widetilde{Y}_{w_j}^T \cdot \widetilde{Z}_{w_{it}}$ is equal to 0. As described in Section IV-B, an equality indicates a match of the queried keyword and the inverted list. We assume that the corresponding inverted lists are $\widehat{\mathbb{I}}_Q = \{\widehat{\mathbb{I}}_{w_i}\}$, where $1 \le i \le k$, i.e., $k = |\widehat{\mathbb{I}}_Q|$. Then, the server identifies the documents containing the exact queried phrase by determining the positional relationship of the keywords using (3) and (4). Finally, the server replies to the user with the search results, i.e., the corresponding encrypted documents that contain the queried phrase.

V. Security Analysis

This section presents the security analysis under the *known ciphertext model* and the *known background model*. We adopt the security definitions in [7].

1) *History:* Let Γ be a file set and $\mathbb{I}$ be the index built from Γ. A *history* over Γ is a tuple $H = (\Gamma, \mathbb{I}, w)$, where w is a phrase containing k keywords $w = (w_1, w_2, \ldots, w_k)$.
2) *View*, denoted by $V(H)$, is the encrypted form of H under a certain secret key sk. In general, a $V(H)$ consists of the encrypted documents $Enc_{sk}(\Gamma)$, the secure index $Enc_{sk}(\mathbb{I}(\Gamma))$, and the secure trapdoor $Enc_{sk}(w)$. Note that the cloud server can only know the views.
3) *Trace:* The trace of history, which is denoted by $Tr(H)$, consists of exactly the information we are willing to leak about the history and nothing else. More precisely, it should be the access patterns and the search results induced by H. The trace induced by a *history* $H = (\Gamma, \mathbb{I}, w)$, is a sequence $Tr(H) = Tr(w) = \{R_w, (\delta_i)_{w \subset \delta_i}, 1 \le i \le |\Gamma|\}$, where w should occur in the document δ_i as a phrase, and R_w indicates whether these keywords constitute a phrase in the documents.

Theorem 1: Our phrase search scheme is secure under the known ciphertext model.

Intuitively, given two histories with the same trace, if the cloud server cannot distinguish which one is generated by a simulator, we can say that it cannot learn additional information about the secure index or the encrypted documents, except for the access patterns and search results.

Proof: Assume that S is a simulator that can simulate a view V' indistinguishable from the view obtained by the cloud server. To achieve this, we construct the simulator as follows.

1) S selects a random $\delta_i' \in \{0,1\}^{|\delta_i|}$, $\delta_i \in \Gamma$, $1 \le i \le |\Gamma|$, and then outputs $\Gamma' = \{\delta_i', 1 \le i \le |\Gamma'|\}$.
2) S first generates two random τ-bit big primes p' and q' to obtain $n' = p' * q'$, and constructs the bilinear groups $\mathbb{G}'$ and $\mathbb{G}_T'$. Then, S selects two random generators g' and u' from $\mathbb{G}'$ and obtains $h' = u'^{q'}$. Finally, S randomly picks a d-bit binary string S', two $d \times d$ invertible matrices M_1', M_2', a secure hash function $\pi(\cdot)$ with a secret key K', and a secure PRP primitive ν with the secret key U'. Let $sk' = \{p', g', h', K', U', S', M_1', M_2'\}$.
3) S generates $\mathbb{I}'(\Gamma')$ with the same dictionary W as Γ. For each $w_i \in W$, S takes the following steps.
   a) S picks a random binary string as the inverted list $\mathbb{I}'_{w_i}$, which has the same length as the actual inverted list $\mathbb{I}_{w_i}$. Ensure that if $w_i \in W$ and $w_i \subset \delta_i$, $1 \le i \le |\Gamma|$, the inverted list $\mathbb{I}'_{w_i}$ should contain the identifier $\nu(U', id(\delta_i))$ of δ_i. Meanwhile, if w occurs in δ_i as a phrase, we should also ensure that w occurs in δ_i' as a phrase.
   b) S gets $\widetilde{B}' = \{\pi(K', w_i)^0, \ldots, \pi(K', w_i)^{d-1}\}^T$ and computes $Enc_{sk'}(\widetilde{B}')$. Finally, S obtains $Enc_{sk'}(\mathbb{I}'(\Gamma'))$.
4) S constructs the query w' and the corresponding trapdoor as follows. For each $w_i \in w$, S constructs the encrypted keyword identifier $\widetilde{Y}_{w_j}$ by Algorithm 2. Then S

sets $Enc_{sk'}(w') = \{\{\widetilde{Y}_{w_j}\}, Enc_{sk'}(1), \lambda'^{p'}\}$ as the trapdoor, where λ' is a random element of $\mathbb{G}'$ and $1 \leq j \leq |w'|$.

5) Finally, S outputs the view $V' = (\Gamma', Enc_{sk'}(I'(\Gamma')), Enc_{sk'}(w'))$.

The correctness of the construction is easy to demonstrate, as the secure index $Enc_{sk'}(I'(\Gamma'))$ and the trapdoor $Enc_{sk'}(w')$ generate the same trace as the one obtained by the cloud server. Hence, we can claim that for any probabilistic polynomial-time (P.P.T.) adversary, V' cannot be distinguished from $V(H)$. Furthermore, no P.P.T. adversary can distinguish the Γ' from $Enc_{sk}(\Gamma)$ for the semantic security of the symmetric encryption. The indistinguishability of the index and trapdoors are guaranteed and enhanced together by the indistinguishability of the secure kNN technique, the random number introduced in the splitting process, and the use of probabilistic encryption. ■

Theorem 2: Our phrase search scheme is secure under the known background model.

Intuitively, given a view generated by the simulator, if the cloud server, who has several pairs of queried phrases and trapdoors, cannot distinguish it from the view he owns, we can say that the proposed phrase search scheme is secure under the known background model.

Proof: Based on the above construction, we can claim that no P.P.T. adversary can distinguish the view V' from $V(H)$ with a certain number of pairs of keywords and trapdoors. Particularly, no P.P.T. adversary can distinguish the Γ' from $Enc_{sk}(\Gamma)$ for the semantic security of the symmetric encryption. Due to the usage of the dummy keywords and the probabilistic encryption, the same queries will have different trapdoors. Therefore, the P.P.T. adversary cannot launch the linear analysis using the pairs of queried phrases and trapdoors. Thus, the indistinguishability of indices and trapdoors are guaranteed. ■

VI. Performance Evaluation

In this section, we evaluate the performance of P3 through extensive experiments using real-world datasets.

A. Experiment Setup

Testbed: To simulate the cloud-based service environment, we use an Aliyun server instance[2] as the cloud server, which is equipped with an Intel Xeon processor at 2.60 GHz and 8 GB RAM.

Dataset: We use a collection of the requests for comments (RFCs) [24] as the real-world dataset for evaluation. Each file contains a large number of technical phrases, e.g., *error detection*. We randomly pick up 2500 files from the publicly available RFCs. For each file in the dataset, we build a full-text index, which is the same as the one commonly used by modern search engines.

Methods to Compare: We compare P3 with a representative phrase search solution [30] and the traditional multikeyword conjunctive search scheme, which are referred to as *PSSE* and *conjunctive search*, respectively. Since an implementation of PSSE is not given in [30], we implement it using Java. The conjunctive search scheme can be implemented simply by ignoring the phrase recognition procedure in P3. Although it is not an exact implementation of an existing solution in the literature, it can still help us to understand the differences of the results returned by the conjunctive multikeyword search and the phrase search. We use a 128-bit security parameter in all the three methods.

[2][Online]. Available: https://www.aliyun.com/

TABLE III
Summary of Index Construction Overheads

Metrics / Methods		Number of Documents				
		500	1000	1500	2000	2500
Time (h)	PSSE	0.23	2.9	5.4	8.3	11.8
	P3	0.64	3.5	7.1	11.2	15.1
Volume (GB)	PSSE	1.48	19.9	35.9	57.5	78.3
	P3	0.08	0.31	0.56	0.87	1.15

The threshold parameter η is set to be 32, which is as the frequency median of all keywords in the document set. We denote $|Q|$ as the phrase length (i.e., the number of keywords in the phrase) and m as the number of documents.

Query Sets: We generate the querying phrases by randomly choosing phrases with semantics from the file set, e.g., sophisticated terminals, interrupt characters, shared memory, etc. We use the same query length setting as existing studies [1], where $|Q|$ takes the concrete values of 2, 3, 4, and 5.

B. Search Accuracy

We adopt a definition of the search accuracy widely used in [32]. Given a phrase query, the search accuracy $\mathcal{P}$ is calculated as $\mathcal{P} = t_p/(f_p + t_p)$, where t_p and f_p are the numbers of relevant (i.e., containing the exact phrase) and irrelevant (i.e., containing all the keywords rather than the exact phrase) documents in the search results.

We first fix $|Q| = 2$ and explore the numbers of matched documents for each method with varying scales of the document set, as shown in Fig. 4(a). Compared with the conjunctive search scheme, P3 and PSSE can remarkably reduce the number of matched documents.

The precision with respect to different query lengths for each method is depicted in Fig. 4(b). Here, the plain index phrase search scheme serves as the baseline of the precision. We can see that the precisions of P3 and PSSE are 100% in all the cases, whereas those of the conjunctive search scheme are less than 20% in all the cases.

C. Search Efficiency

Index Construction: The index construction process is a one-time, offline computation. The time and storage overheads of the index construction are depicted in Table III. Clearly, the overheads increase when the document set gets larger. For the same document set, the index size of PSSE is much larger than that of P3. As to the index construction time, P3 requires slightly more time than PSSE, which is primarily caused by the encryption operations of the keyword locations.

Trapdoor Generation: The trapdoor generation time for each method with different query lengths is depicted in Fig. 5.

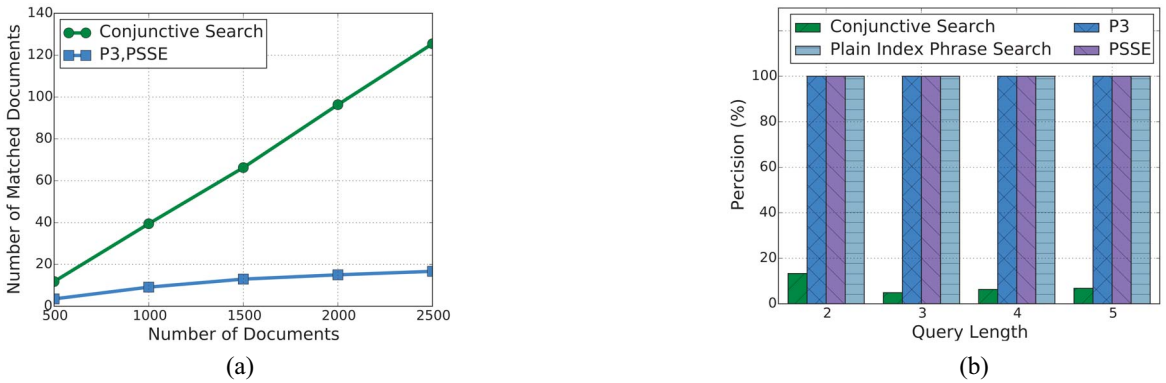

Fig. 4. Search accuracy with varying document sets and query lengths. (a) Number of documents in search results ($|Q| = 2$). (b) Precision with different query lengths ($m = 2500$).

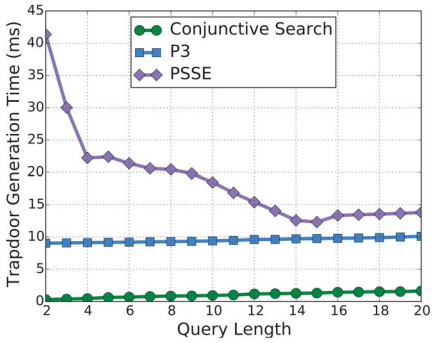

Fig. 5. Trapdoor generation time for different query lengths ($m = 2500$).

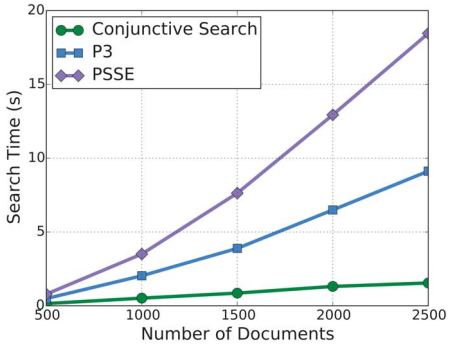

Fig. 7. Search time for different numbers of indexed documents ($|Q| = 2$).

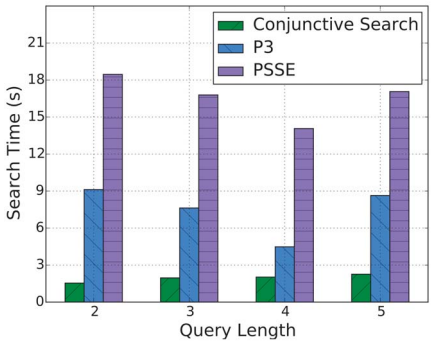

Fig. 6. Search time for different query lengths ($m = 2500$).

P3 has a higher time cost of trapdoor generation than the conjunctive search scheme, because it needs extra operations (e.g., generating dispersal factor) to generate additional information for phrase judgement. Compared with PSSE, P3 can reduce the time cost, especially when the query length is less than 8. This is because PSSE needs two rounds of interaction between the user and the cloud server, and during the second interaction, it needs to generate a trapdoor for each document that was returned in the first interaction. As the query length increases, the number of documents returned in the first interaction could drop, which leads to a fall of the trapdoor generation time for PSSE.

Query Time: The query time is defined as the time interval from the submission of a user's trapdoor to the receival of the search results. For each queried phrase, we repeat the query 20 times and calculate the average search time to mitigate the deviation caused by uncertain factors. Note that PSSE may result in a huge index size (see Table III), which cannot be loaded completely into the memory used in our experiments. Therefore, we enable the query algorithms of P3 and PSSE to dynamically load the partial index.

Fig. 6 shows the relationship between the search time and the query length. The conjunctive search scheme takes the shortest search time. However, such a scheme cannot provide accuracy guarantees as discussed in Section VI-B. As to the phrase search schemes, P3 can roughly reduce the average search time of PSSE by half. This is because PSSE has a large index size and thereby spends more time than P3 on loading its index into the memory.

The search time with different document scales is shown in Fig. 7. Here, we exhibit only the results for $|Q| = 2$ due to space limitation. The search time for each of the three methods enlarges with the growth of the number of documents. Compared with PSSE, P3 can greatly reduce the average search time for different scales of document sets.

Communication Overhead: The communication time and data volumes are depicted in Fig. 8. The communication time means the transmission time of the trapdoors and search results between the client and the cloud server. As the number of indexed documents grows, the communication time becomes higher for all three methods. In particular, P3 has the shortest communication time, because P3 has the smallest data

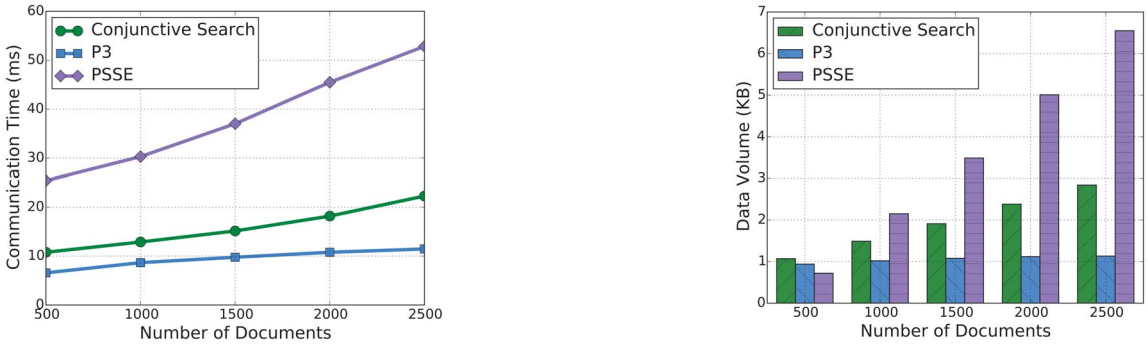

Fig. 8. Communication overhead with different numbers of documents ($|Q| = 2$). (a) Communication time. (b) Communication data volumes.

volume. First, P3 has a higher search accuracy than the conjunctive search scheme, and thereby gets a smaller volume of search results that should be replied from the cloud server to the client. Second, compared with PSSE, P3 only needs one-round of interaction and avoids sending intermediate data to the client for phrase recognition.

VII. Discussion

Although the proposed scheme is more efficient than the existing phrase search schemes, there are still two limitations.

First, compared with the conjunctive search scheme, P3 has to spend more time on phrase recognition, and thereby increases search time. Second, P3 cannot directly support a flexible index update due to the inherent feature of the inverted index and the adoption of the padding strategy.

A possible way to mitigate these limitations is leveraging the parallel processing techniques over server clusters. We can partition the whole document set into several subsets, each of which contain partial documents and is indexed independently. Given a phrase query from the user, search operations can be performed in parallel over the subsets, which helps to shorten the search time. An offline update of the secure index can be employed to deal with updates, e.g., add or remove of documents and keywords. In particular, when a document has to be updated, we only need to regenerate the index of the corresponding subset which the document belongs to, thereby reducing the index update overhead. We leave these attempts for the future work.

VIII. Conclusion

In this paper, we presented a novel scheme, P3, which tackled the challenges in phrase search for intelligent encrypted data processing in cloud-based IoT. The scheme exploits the homomorphic encryption and bilinear map to determine the pairwise location relationship of queried keywords on the cloud server side. It eliminates the need of a trusted third party and greatly reduces communication overheads. Thorough security analysis illustrated that the proposed scheme provides the desired security guarantees. The experimental evaluation results demonstrated the effectiveness and efficiency of the proposed scheme. In future work, we plan to further improve the flexibility and efficiency of the scheme.

References

[1] A. Anand, I. Mele, S. Bedathur, and K. Berberich, "Phrase query optimization on inverted indexes," in *Proc. ACM CIKM*, 2014, pp. 1807–1810.

[2] S. Ananthi, M. S. Sendil, and S. Karthik, "Privacy preserving keyword search over encrypted cloud data," *Commun. Comput. Inf. Sci.*, vol. 190, pp. 480–487, 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-22709-7_47

[3] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. TCC*, 2005, pp. 325–341.

[4] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.

[5] C. Chen *et al.*, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[6] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. Workshops IEEE ICDCS*, Jun. 2011, pp. 273–281.

[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM CCS*, Alexandria, VA, USA, 2006, pp. 79–88.

[8] B. Dan, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, 2004, pp. 506–522.

[9] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Transactions papers a routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1223–1229, Mar. 2009.

[10] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 24–34, 2007.

[11] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2017.

[12] F. Gao *et al.*, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Netw.*, to be published, doi: 10.1109/MNET.2018.1700269.

[13] X. Hei, X. Du, S. Lin, and I. Lee, "PIPAC: Patient infusion pattern based access control scheme for wireless insulin pump system," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 3030–3038.

[14] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM CCS*, New York, NY, USA, 2012, pp. 965–976.

[15] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.

[16] H. Li *et al.*, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 3, pp. 312–325, May/Jun. 2016.

[17] H. Li *et al.*, "Blockchain-based data preservation system for medical data," *J. Med. Syst.*, vol. 42, no. 8, p. 141, Jun. 2018.

[18] J. Li *et al.*, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5.
[19] Y. Liu, Z. Li, W. Guo, and W. Chaoxia, "Privacy-preserving multi-keyword ranked search over encrypted big data," in *Proc. Int. Conf. Cyberspace Technol.*, 2016, pp. 1–3.
[20] H. T. Poon and A. Miri, "A low storage phase search scheme based on bloom filters for encrypted cloud services," in *Proc. IEEE 2nd Int. Conf. Cyber Security Cloud Comput.*, New York, NY, USA, Nov. 2015, pp. 253–259.
[21] H. Raviv, O. Kurland, and D. Carmel, "The cluster hypothesis for entity oriented search," in *Proc. ACM SIGIR*, New York, NY, USA, 2013, pp. 841–844.
[22] Y. Xiao *et al.*, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, nos. 11–12, pp. 2314–2341, 2007.
[23] Y. Ren, Y. Chen, J. Yang, and B. Xie, "Privacy-preserving ranked multi-keyword search leveraging polynomial function in cloud computing," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 594–600.
[24] RFC. *Request for Comments Database*. Accessed: May 1, 2016. [Online]. Available: http://www.ietf.org/rfc.html
[25] M. Shen, G. Cheng, L. Zhu, X. Du, and J. Hu, "Content-based multi-source encrypted image retrieval in clouds with privacy preservation," *Future Gener. Comput. Syst.*, May 2018, doi: 10.1016/j.future.2018.04.089.
[26] M. Shen *et al.*, "Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 940–953, Apr. 2018.
[27] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1830–1843, Aug. 2017.
[28] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE S&P*, 2000, pp. 44–55.
[29] W. Sun *et al.*, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. ASIACCS*, New York, NY, USA, 2013, pp. 71–82.
[30] Y. Tang, D. Gu, N. Ding, and H. Lu, "Phrase search over encrypted data with symmetric encryption scheme," in *Proc. Workshops IEEE ICDCS*, Jun. 2012, pp. 471–480.
[31] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 2092–2100.
[32] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 2112–2120.
[33] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE ICDCS*, Jun. 2010, pp. 253–262.
[34] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
[35] C. Yang, W. Zhang, J. Xu, J. Xu, and N. Yu, "A fast privacy-preserving multi-keyword search scheme on cloud data," in *Proc. Int. Conf. Cloud Service Comput.*, 2013, pp. 104–110.
[36] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu, "Prometheus: Privacy-aware data retrieval on hybrid cloud," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2643–2651.
[37] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 628–643, Mar. 2018.
[38] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in *Proc. IEEE GLOBECOM*, Dec. 2012, pp. 764–770.

Meng Shen (M'14) received the B.Eng. degree in computer science from Shandong University, Jinan, China, in 2009, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2014.

He is currently with the Beijing Institute of Technology, Beijing, as an Associate Professor. His current research interests include privacy protection for cloud and IoT, blockchain applications, and encrypted traffic classification.

Dr. Shen was a recipient of the Best Paper Runner-Up Award of IEEE IPCCC 2014.

Baoli Ma received the B.Eng. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2015, where he is currently pursuing the M.S. degree at the Department of Computer Science.

His current research interests include cloud computing and secure searchable encryption.

Liehuang Zhu (M'11) is a Professor with the Department of Computer Science, Beijing Institute of Technology, Beijing, China. He was selected into the Program for New Century Excellent Talents in University from Ministry of Education, Beijing. His current research interests include Internet of Things, cloud computing security, and Internet and mobile security.

Xiaojiang Du (M'04–SM'09) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 2002 and 2003, respectively.

He is a tenured Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. He authored a book published by Springer. He has been awarded over $5 million research grants from the U.S. National Science Foundation, Army Research Office, Air Force, NASA, the State of Pennsylvania, and Amazon. He has authored over 300 journal and conference papers. His current research interests include wireless communications, wireless networks, security, and systems.

Dr. Du was a recipient of the Best Paper Award of IEEE GLOBECOM 2014 and the Best Poster Runner-Up Award of ACM MobiHoc 2014. He serves on the Editorial Boards of three international journals. He is a Life Member of the ACM.

Ke Xu (M'02–SM'09) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China.

He serves as a Full Professor with Tsinghua University. He is currently a Visiting Professor with the University of Essex, Colchester, U.K. He has authored or co-authored over 100 technical papers and holds 20 patents. His current research interests include next generation Internet, P2P systems, Internet of Things, and network virtualization and optimization.

Dr. Xu has guest edited several special issues in IEEE and Springer journals. He is a member of the ACM.